

BBBBBBBBBBBB		AAAAAAA		SSSSSSSSSS		RRRRRRRRRR		TTTTTTTTTTTT		LLL
BBBBBBBBBBBB		AAAAAAA		SSSSSSSSSS		RRRRRRRRRR		TTTTTTTTTTTT		LLL
BBBBBBBBBBBB		AAAAAAA		SSSSSSSSSS		RRRRRRRRRR		TTTTTTTTTTTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRRRRRRRRR		TTT		LLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRRRRRRRRR		TTT		LLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRRRRRRRRR		TTT		LLL
BBB	BBB	AAAAAAAAAAAA			SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAAAAAAAAAAA			SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAAAAAAAAAAA			SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA		SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA		SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA		SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA		SSS	RRR	RRR	TTT		LLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRR	RRR	TTT		LLLLLLLLLLLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRR	RRR	TTT		LLLLLLLLLLLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRR	RRR	TTT		LLLLLLLLLLLL

```
BBBBBBBBB  AAAAAA  SSSSSSSS  RRRRRRRR  EEEEEEEEE  CCCCCCCC  PPPPPPPP  RRRRRRRR  000000
BBBBBBBBB  AAAAAA  SSSSSSSS  RRRRRRRR  EEEEEEEEE  CCCCCCCC  PPPPPPPP  RRRRRRRR  000000
BB      BB  AA      AA  SS      SS  RR      RR  EE      EE  CC      CC  PP      PP  RR      RR  00      00
BB      BB  AA      AA  SS      SS  RR      RR  EE      EE  CC      CC  PP      PP  RR      RR  00      00
BB      BB  AA      AA  SS      SS  RR      RR  EE      EE  CC      CC  PP      PP  RR      RR  00      00
BB      BB  AA      AA  SS      SS  RR      RR  EE      EE  CC      CC  PP      PP  RR      RR  00      00
BBBBBBBBB  AA      AA  SSSSSS  RRRRRRRR  EEEEEEE  CC      CC  PPPPPPP  RRRRRRRR  00      00
BBBBBBBBB  AA      AA  SSSSSS  RRRRRRRR  EEEEEEE  CC      CC  PPPPPPP  RRRRRRRR  00      00
BB      BB  AAAAAAAAAA  SS      SS  RR      RR  EE      EE  CC      CC  PP      PP  RR      RR  00      00
BB      BB  AAAAAAAAAA  SS      SS  RR      RR  EE      EE  CC      CC  PP      PP  RR      RR  00      00
BB      BB  AA      AA  SS      SS  RR      RR  EE      EE  CC      CC  PP      PP  RR      RR  00      00
BB      BB  AA      AA  SSSSSSSS  RR      RR  EE      EE  CCCCCCCC  PP      PP  RR      RR  000000
BBBBBBBBB  AA      AA  SSSSSSSS  RR      RR  EEEEEEEEE  CCCCCCCC  PP      PP  RR      RR  000000
```

....  
....  
....  
....

```
LL      LL      SSSSSSSS
LL      LL      SSSSSSSS
LL      LL      SS
LL      LL      SS
LL      LL      SS
LL      LL      SSSSSS
LL      LL      SSSSSS
LL      LL      SS
LL      LL      SS
LL      LL      SS
LL      LL      SS
LLLLLLLLLL  IIIIII  SSSSSSSS
LLLLLLLLLL  IIIIII  SSSSSSSS
```



! Record processing level of abstraction  
! File: BASRECPRO.B32 Edit:MDL1095

```
1 0001 0 MODULE BAS$$REC_PROC (  
2 0002 0 IDENT = '1-095'  
3 0003 0 ) =  
4 0004 1 BEGIN  
5 0005 1  
6 0006 1 *****  
7 0007 1 *  
8 0008 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY  
9 0009 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.  
10 0010 1 * ALL RIGHTS RESERVED.  
11 0011 1 *  
12 0012 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  
13 0013 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE  
14 0014 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER  
15 0015 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  
16 0016 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY  
17 0017 1 * TRANSFERRED.  
18 0018 1 *  
19 0019 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE  
20 0020 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT  
21 0021 1 * CORPORATION.  
22 0022 1 *  
23 0023 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS  
24 0024 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.  
25 0025 1 *  
26 0026 1 *****  
27 0027 1  
28 0028 1  
29 0029 1 ++  
30 0030 1 FACILITY: BASIC Support Library - not user callable  
31 0031 1  
32 0032 1 ABSTRACT:  
33 0033 1  
34 0034 1 This module implements the record processing level of  
35 0035 1 abstraction which is the 3rd level and is called only from  
36 0036 1 the user data formatter level (2nd level) when the user  
37 0037 1 portion of a record buffer is full (WRITE) or empty  
38 0038 1 (READ). This module adds any per record formatting (as  
39 0039 1 distinguished from per I/O statement or per I/O list element  
40 0040 1 formatting) and then calls RMS ($PUT or $GET). RMS errors  
41 0041 1 are converted to BASIC errors and are signaled.  
42 0042 1  
43 0043 1 ENVIRONMENT: User access mode; AST level or not.  
44 0044 1  
45 0045 1 AUTHOR: Donald G. Petersen; CREATION DATE: 16-Mar-78  
46 0046 1  
47 0047 1 MODIFIED BY:  
48 0048 1  
49 0049 1 0-61 - Add ATTEMPT TO READ NON-EXISTANT RECORD error to seq. reads.  
50 0050 1 JMT 02-Jan-78  
51 0051 1 Donald G. Petersen, 16-Mar-78 : VERSION 1-01  
52 0052 1 1-01 - original DGP  
53 0053 1 1-02 - Change to JSB linkages. DGP 14-Nov-78  
54 0054 1 1-004 - Update copyright notice and add device names to REQUIRE  
55 0055 1 files. JBS 29-NOV-78  
56 0056 1 1-005 - Add BAS$RECOUNT to support the Basic RECOUNT function. DGP  
57 0057 1 03-Dec-78
```

58	0058	1	1-006	- Dot problem in BASS\$RECOUNT. DGP 04-Dec-78
59	0059	1	1-007	- Add fudge factor to RECOUNT for the line terminator if input is from a terminal. DGP 04-Dec-78
60	0060	1		
61	0061	1	1-008	- Change REQUIRE file names from FOR... to OTS... JBS 07-DEC-78
62	0062	1	1-009	- Call BASS\$SIGNAL_IO for RMS errors. JBS 18-DEC-78
63	0063	1	1-010	- Add new routines for READ. DGP 19-Dec-78
64	0064	1	1-011	- Change references to ISB\$A_BUF_PTR, BUF_BEG, BUF_END to LUB. DGP 05-Jan-79
65	0065	1		
66	0066	1	1-012	- Change to CR format for terminal data files. DGP 11-Jan-79
67	0067	1	1-013	- Make a few changes for recursive I/O. DGP 15-Jan-79
68	0068	1	1-014	- Remove signalling for ^Z. Moved to UDF level for INPUT LINE handling. DGP 15-Jan-79
69	0069	1		
70	0070	1	1-015	- Change stack frame prefix to BSF\$. JBS 08-FEB-1979
71	0071	1	1-016	- Add BASS\$REC_GSE (Basic GET sequential). DGP 19-Feb-79
72	0072	1	1-017	- Add BASS\$REC_PSE (Basic PUT sequential). DGP 20-Feb-79
73	0073	1	1-018	- set RAB\$R_RBF in BASS\$REC_PSE. DGP 20-Feb-79
74	0074	1	1-019	- Set RAB\$R_RBF in BASS\$GSE. DGP 21-Feb-79
75	0075	1	1-020	- Null fill buffer for GET. DGP 27-Feb-79
76	0076	1	1-021	- Add REC routines for FIND, DELETE, UPDATE, RESTORE, SCRATCH. DGP 27-Feb-79
77	0077	1		
78	0078	1	1-022	- Add BASIOERR.REQ for error handling. DGP 28-Feb-79
79	0079	1	1-023	- Add BASUNLOCK and BASFREE. DGP 28-Feb-79
80	0080	1	1-024	- Set RAB\$R_RBF in BASS\$REC_UPD. DGP 01-Mar-79
81	0081	1	1-025	- Add REC_PRE, REC_GRE, REC_FRE. DGP 02-Mar-79
82	0082	1	1-026	- More work on relative I/O. DGP 05-Mar-79
83	0083	1	1-027	- Update pointer for READ in Basic Major Frame in RMF9. DGP 06-Mar-79
84	0084	1	1-028	- Add support for Basic "foreign buffers". DGP 27-Mar-79
85	0085	1	1-029	- Point all GETs and PUTs off to GET_ERROR or PUT_ERROR. DGP 02-Apr-79
86	0086	1	1-030	- Add more routines to support ISAM. DGP 03-Apr-79
87	0087	1	1-031	- Fix PUT sequential to support ISAM. DGP 04-Apr-79
88	0088	1	1-032	- Put in indexed I/O stuff. 06-Apr-79
89	0089	1	1-033	- Bug fixes in indexed. 10-Apr-79 DGP
90	0090	1	1-034	- Implement the WAIT statement, using LUB\$WAIT_TIME. JBS 10-APR-1979
91	0091	1		
92	0092	1	1-035	- Implement the ECHO and NOECHO functions, using LUB\$V_NOECHO. JBS 17-APR-1979
93	0093	1		
94	0094	1	1-036	- Add code to handle single-character input from GET SEQUENTIAL. JBS 17-APR-1979
95	0095	1		
96	0096	1	1-037	- Implement the CTRL0 and RCTRL0 functions, using LUB\$V_CCO. JBS 19-APR-1979
97	0097	1		
98	0098	1	1-038	- Implement the Cancel Typeahead function, using LUB\$V_PTA. JBS 01-MAY-1979
99	0099	1		
100	0100	1	1-039	- Add Basic PRINT USING support. DGP 15-May-79
101	0101	1	1-040	- Change BIND to GLOBAL BIND ROUTINE in PRINT USING support. JBS 16-MAY-1979
102	0102	1		
103	0103	1	1-041	- Add BASS\$RECOU_INIT. JBS 04-JUN-1979
104	0104	1	1-042	- Add REC level for MAT INPUT. DGP 05-Jun-79
105	0105	1	1-043	- Clean up a lot and put real code into Matrix Input routines. DGP 14-Jun-79
106	0106	1		
107	0107	1	1-044	- Make REC_MIN1 look for continuation character. DGP 20-Jun-79
108	0108	1	1-045	- Terminal devices use PRN format for output. DGP 10-Jul-79
109	0109	1	1-046	- Add BASS\$NUM_INIT, BASS\$NUM2_INIT, BASS\$MAT_LINPUT, BASS\$MAT_READ, BASS\$NUM, BASS\$NUM. DGP 13-Jul-79
110	0110	1		
111	0111	1	1-047	- Change ISB\$R_MAJ_F_PTR to ISB\$A_MAJ_F_PTR. JBS 24-JUL-1979
112	0112	1	1-048	- Signal if READ with no DATA. DGP 07-Aug-79
113	0113	1	1-049	- Debug MAT I/O. DGP 07-Aug-79
114	0114	1	1-050	- STOP a few errors that are being SIGNALLed. DGP 05-Sep-79



115	0115	1	1-051	- FREE and UNLOCK are noops if no record locked. DGP 06-Sep-79
116	0116	1	1-052	- Move NUM_INIT and NUM2_INIT to BASS\$MAT_IO, and move BASS\$BLNK_LINE from BASS\$MAT_IO to here. DGP 06-Sep-79
117	0117	1		
118	0118	1	1-053	- Load LUB\$A_RBUF_ADR for GET and PUT for Locate mode (RMS). DGP 13-Sep-79
119	0119	1		
120	0120	1	1-054	- Clear the prompt buffer in GET_ERROR. DGP 17-Sep-79
121	0121	1	1-055	- Fix BASS\$BLNK_LINE. DGP 04-Oct-79
122	0122	1	1-056	- Add MAT_READ. DGP 11-Oct-79
123	0123	1	1-057	- Add a REC9 routine for MAT_PRINT. DGP 12-Oct-79
124	0124	1	1-058	- Add BASS\$REC_MLI1. DGP 12-Oct-79
125	0125	1	1-059	- Fix BASS\$REC_WSL1 to leave the cursor alone. DGP 02-Nov-79
126	0126	1	1-060	- Allow BASS\$REC_WSL1 to accept an argument. DGP 06-Nov-79
127	0127	1	1-061	- GET will only null fill the buffer, if necessary, after a GET. DGP 12-Nov-79
128	0128	1		
129	0129	1	1-062	- BASS\$REC_MPR1 needs an LF if no format char. DGP 13-Nov-79
130	0130	1	1-063	- Use LUB\$A_UBF to simplify foreign buffer code. JBS 13-NOV-1979
131	0131	1	1-064	- BASS\$REC_MIN1 should not differentiate between terminal & non-terminal devices. DGP 14-Nov-79
132	0132	1		
133	0133	1	1-065	- GET relative not null filling the buffer properly. DGP 29-Nov-79
134	0134	1	1-066	- Null fill the buffer before restoring foreign buffer pointers for GET. DGP 18-Dec-79
135	0135	1		
136	0136	1	1-067	- RMS does not return a terminator in the STV field for files. DGP 03-Jan-80
137	0137	1		
138	0138	1	1-068	- REC_WSL9 should only write a record if the output buffer has something in it to write. DGP 03-Jan-80
139	0139	1		
140	0140	1	1-069	- Addition to 1-068. Should also write a record if there was no element transmitter. DGP 04-Jan-80
141	0141	1		
142	0142	1	1-070	- Unconditionally write a CR in WSL1. DGP 14-Jan-80
143	0143	1	1-071	- Restore "foreign buffers" properly and set RECOUNT in GET Indexed and Relative. DGP 12-Feb-80
144	0144	1		
145	0145	1	1-072	- Adjust the Global RECOUNT to include the length of an escape sequence. DGP 22-Feb-80
146	0146	1		
147	0147	1	1-073	- A previous edit to fix a problem with foreign buffers reintroduced a problem with only null padding the buffer after a successful GET. DGP 26-Feb-80
148	0148	1		
149	0149	1		
150	0150	1	1-074	- REC_WSL9 should set VFC2 to BASS\$K_NULL (no carriage control) if there is a format character. DGP 26-Feb-80
151	0151	1		
152	0152	1	1-075	- Update the cursor position for INPUT if terminated by an escape. DGP 27-Feb-80
153	0153	1		
154	0154	1	1-076	- When calculating CCPOS (current cursor position) following an INPUT statement, take the prompt string into account.
155	0155	1		
156	0156	1	1-077	- REC_RSL1 is not updating the cursor position correctly. DGP 04-Mar-80
157	0157	1	1-078	- REC_WSL9 should set the 'pre' carriage control for the next record to [F if there is no format character 'cuz of recursive I/O. DGP 07-Mar-80
158	0158	1		
159	0159	1		
160	0160	1	1-079	- Rationalize the CCO and PTA bits. CCO is now copied from LUB to RAB when initializing for output, and PTA when initializing for input. JBS 31-MAR-1980
161	0161	1		
162	0162	1		
163	0163	1	1-080	- Clear the dirty bit CCB [LUB\$V_OUTBUF_DR] in PUT_ERROR so BASS\$CLOSE when invoked by the unwind won't get confused and do a PUT. FM 11-SEP-80
164	0164	1		
165	0165	1		
166	0166	1	1-081	- Tack on the terminator(s) to the buffer when a GET is done on a terminal device file in BASS\$REC_GSE.
167	0167	1		
168	0168	1	1-082	- Add/transfer BASS\$WAIT to this module, now wait routines are part of the sharable image. The routines added are BASS\$WAIT, BASS\$READ_WAIT. We had to make WAIT routines part of the sharable image because WAIT was requested to become a GLOBAL, and routines in
169	0169	1		
170	0170	1		
171	0171	1		



```

: 172      0172 1  this module had to read it.
: 173      0173 1  1-083- Only if LUB$B_RAT indicates CR format tack on the CRLF. FM 9-feb-81
: 174      0174 1  1-084- Cursor position not updated correctly if INPUT was
: 175      0175 1  terminated by an escape - code should check if previous
: 176      0176 1  PRINT was terminated by a semicolon or comma. PLL 5-7-81
: 177      0177 1  1-085- The purge typeahead function is no longer setting the PTA bit in
: 178      0178 1  the LUB, so BAS$$REC_RSLO, BAS$$REC_MINO, and BAS$$REC_GSE don't need
: 179      0179 1  to check it anymore. PLL 6-Aug-81
: 180      0180 1  1-086 - Add support for RFA access and manual record locking. PLL 1-Jun-82
: 181      0181 1  1-087 - BAS$$REC_GIN and BAS$$REC_FIN should check for a decimal key
: 182      0182 1  when setting the key size in the RAB. PLL 6-Jul-1982
: 183      0183 1  1-088 - Add support for ANSI INPUT. If not enough data is supplied, the
: 184      0184 1  entire INPUT must be restarted. PLL 29-Jul-1982
: 185      0185 1  1-089 - Fix CTRL/O rationalization. Unconditionally copy whatever its
: 186      0186 1  state is in the LUB into the RAB, and clear it in the LUB.
: 187      0187 1  MDL and JBS 10-Aug-1982
: 188      0188 1  1-090 - Set buffer pointer (in WSL9) from buffer beginning pointer rather
: 189      0189 1  than from RBUF_ADR. This fixes the problem of pointing to the
: 190      0190 1  wrong buffer when the user enters a CTRL/C while a line is being
: 191      0191 1  written, and his control-c routine writes a line also.
: 192      0192 1  MDL and PLL 19-Aug-1982
: 193      0193 1  1-091 - In BAS$$REC_RSL1, BAS$$SIGNAL_IO the too little data error instead
: 194      0194 1  of calling BAS$$STOP_IO. PLL 27-Sep-1982
: 195      0195 1  1-092 - In BAS$$REC_RSLO, the contents of the print buffer should be $PUT
: 196      0196 1  before the $GET is done, if this is a non-terminal device. This
: 197      0197 1  is a requested behavior change that will cause input prompts to
: 198      0198 1  appear in batch log files. RMS is making a change concurrently
: 199      0199 1  that will cause the actual input provided to appear in the batch
: 200      0200 1  log as well, thus making a batch log an exact duplicate of how
: 201      0201 1  it would appear if run interactively. MDL 26-Jul-1983
: 202      0202 1  1-093 - check for RMS$_CONTROL_C completion status; call new CTRL/C signaller
: 203      0203 1  if this status is returned, this change is coordinated with rev.
: 204      0204 1  2-003 of BAS$CTRLC. MDL 12-Mar-1984
: 205      0205 1  1-094 - for the special case of channel 0, edit 1-092 should reach thru the
: 206      0206 1  buddy ptr and use the output side of channel 0 to write out the
: 207      0207 1  prompt string. MDL 22-Mar-1984
: 208      0208 1  1-095 - routines that set and reset record options should reset them BEFORE
: 209      0209 1  calling error routines, so that subsequent I/O on the channel will
: 210      0210 1  work properly (if the user handles the error). MDL 23-Mar-1984
: 211      0211 1  --
: 212      0212 1
: 213      0213 1  !<BLF/PAGE>

```



```
215 0214 1 |
216 0215 1 | SWITCHES:
217 0216 1 |
218 0217 1 |
219 0218 1 | SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
220 0219 1 |
221 0220 1 |
222 0221 1 | LINKAGES
223 0222 1 |
224 0223 1 |
225 0224 1 | REQUIRE 'RTLIN:OTSLNK'; | define all linkages
226 0653 1 |
227 0654 1 |
228 0655 1 | TABLE OF CONTENTS:
229 0656 1 |
230 0657 1 |
231 0658 1 | FORWARD ROUTINE
232 0659 1 | BAS$$WAIT : NOVALUE, | Writes into module level OWN WAIT
233 0660 1 | BAS$$READ_WAIT, | Reads the module level OWN WAIT
234 0661 1 | BAS$$RECOUNT, | Support Basic RECOUNT function
235 0662 1 | BAS$$RECOU_INIT : NOVALUE, | Initialize RECOUNT
236 0663 1 | BAS$$BLNK_LINE : CALL_CCB NOVALUE, | print a blank line
237 0664 1 | ! write sequential list-directed
238 0665 1 | BAS$$REC_WSL0 : JSB_REC0 NOVALUE, | initialize output buffer
239 0666 1 | BAS$$REC_WSL1 : JSB_REC WSL1 NOVALUE, | write all but last record
240 0667 1 | BAS$$REC_WSL9 : JSB_REC9 NOVALUE, | write last record
241 0668 1 | ! Mat Linput
242 0669 1 | BAS$$REC_MLI1 : JSB_REC1, | always read another record
243 0670 1 | ! Mat Read
244 0671 1 | BAS$$REC_MRE1 : JSB_REC1, | return a failure
245 0672 1 | ! Mat Print
246 0673 1 | BAS$$REC_MPR1 : JSB_REC1 NOVALUE, | write one buffer
247 0674 1 | BAS$$REC_MPR9 : JSB_REC9 NOVALUE, | terminate Mat Print
248 0675 1 | ! read sequential list-directed
249 0676 1 | BAS$$REC_RSL0 : JSB_REC0 NOVALUE, | read first
250 0677 1 | BAS$$REC_RSL1 : JSB_REC1, | read next
251 0678 1 | BAS$$REC_RSL9 : JSB_REC9 NOVALUE, | no-op
252 0679 1 | ! MAT INPUT
253 0680 1 | BAS$$REC_MIN0 : JSB_REC0 NOVALUE, | initialize MAT INPUT
254 0681 1 | BAS$$REC_MIN1 : JSB_REC1, | record handler
255 0682 1 | BAS$$REC_MIN9 : JSB_REC9 NOVALUE, | terminate MAT INPUT
256 0683 1 | ! read memory list-directed
257 0684 1 | BAS$$REC_RMF0 : JSB_REC0 NOVALUE, | initialize read memory
258 0685 1 | BAS$$REC_RMF1 : JSB_REC1 NOVALUE, | signal insufficient data
259 0686 1 | BAS$$REC_RMF9 : JSB_REC9 NOVALUE, | no-op
260 0687 1 |
261 0688 1 | GLOBAL BIND
262 0689 1 | ROUTINE
263 0690 1 | ! write formatted
264 0691 1 | BAS$$REC_WF0 = BAS$$REC_WSL0,
265 0692 1 | BAS$$REC_WF1 = BAS$$REC_WSL1,
266 0693 1 | BAS$$REC_WF9 = BAS$$REC_WSL9;
267 0694 1 |
268 0695 1 | FORWARD ROUTINE
269 0696 1 | ! record operations
270 0697 1 | BAS$$REC_GSE : JSB_DO_READ NOVALUE, | GET sequential
271 0698 1 | BAS$$REC_PSE : JSB_PUT NOVALUE, | PUT sequential
```

```
272 0699 1 BASS$REC_FSE : JSB_REC2 NOVALUE,      ! FIND sequential
273 0700 1 BASS$REC_FRFA: JSB_REC2 NOVALUE,      ! FIND by RFA
274 0701 1 BASS$REC_DSE : JSB_REC0 NOVALUE,      ! DELETE sequential
275 0702 1 BASS$REC_UPD : JSB_DO_WRITE NOVALUE,   ! UPDATE
276 0703 1 BASS$REC_RSE : JSB_REC0 NOVALUE,      ! RESTORE sequential
277 0704 1 BASS$REC_SSE : JSB_REC0 NOVALUE,      ! SCRATCH
278 0705 1 BASS$REC_PRE : JSB_PUT NOVALUE,        ! PUT relative with count
279 0706 1 BASS$REC_GRE : JSB_DO_READ NOVALUE,    ! GET relative
280 0707 1 BASS$REC_GRFA: JSB_DO_READ NOVALUE,    ! GET by RFA
281 0708 1 BASS$REC_FRE : JSB_REC0 NOVALUE,      ! FREE relative
282 0709 1 BASS$REC_UNL : JSB_REC0 NOVALUE,      ! UNLOCK
283 0710 1 BASS$REC_FEE : JSB_REC0 NOVALUE,      ! FREE
284 0711 1 BASS$REC_GIN : JSB_REC_IND1 NOVALUE,   ! GET indexed
285 0712 1 BASS$REC_FIN : JSB_REC_IND1 NOVALUE,   ! FIND indexed
286 0713 1 BASS$REC_RIN : JSB_REC_IND NOVALUE,    ! RESTORE indexed
287 0714 1 PUT_ERROR : CALL_CCB NOVALUE,         ! error in $PUT
288 0715 1 GET_ERROR  : CALL_CCB NOVALUE;        ! error in $GET
289 0716 1
290 0717 1
291 0718 1 ! INCLUDE FILES:
292 0719 1 !
293 0720 1
294 0721 1 REQUIRE 'RTLIN:BASIOERR';              ! I/O error codes
295 0774 1
296 0775 1 REQUIRE 'RTLIN:BASFRAME';              ! Basic frame offsets
297 0978 1
298 0979 1 REQUIRE 'RTLML:OTSISB';                ! I/O statement block (ISB) offsets
299 1147 1
300 1148 1 REQUIRE 'RTLML:OTSLUB';                ! Logical unit block (LUB) offsets
301 1288 1
302 1289 1 REQUIRE 'RTLIN:OTSMAC';                ! Macros
303 1483 1
304 1484 1 REQUIRE 'RTLIN:RTLPSECT';              ! Define DECLARE_PSECTS macro
305 1579 1
306 1580 1 REQUIRE 'RTLML:BASPAR';                ! BASIC inter-module parameters
307 1602 1
308 1603 1 LIBRARY 'RTLSTARLE';                  ! STARLET library for macros and symbols
309 1604 1
310 1605 1
311 1606 1 ! MACROS:
312 1607 1
313 1608 1 !
314 1609 1 !
315 1610 1 !
316 1611 1 !
317 1612 1 !
318 1613 1 LITERAL
319 1614 1 K_MAT_CONT_CHAR = %X'26',              ! 'g' - Mat Input continuation
320 1615 1                                         ! character
321 1616 1 K_STOP = 0,                          ! stop after signalling this error
322 1617 1 K_SIGNAL = 1;                        ! signal and allow restart
323 1618 1
324 1619 1 !
325 1620 1 ! PSECT DECLARATIONS:
326 1621 1
327 1622 1 DECLARE_PSECTS (BAS);                  ! declare PSECTS for BASS$ facility
328 1623 1 !
```



```

: 329      1624 1 ! OWN STORAGE:
: 330      1625 1
: 331      1626 1 OWN
: 332      1627 1   RECOUNT : INITIAL (0),
: 333      1628 1   WAIT      : WORD INITIAL (0);
: 334      1629 1
: 335      1630 1 !
: 336      1631 1 ! EXTERNAL REFERENCES:
: 337      1632 1 !
: 338      1633 1 !
: 339      1634 1 EXTERNAL ROUTINE
: 340      1635 1   BAS$$SIGNAL : NOVALUE,
: 341      1636 1   BAS$$STOP_IO : NOVALUE,
: 342      1637 1   BAS$$SIGNAL_IO : NOVALUE,
: 343      1638 1   BAS$$SIGNAL_CTRL : NOVALUE;
: 344      1639 1
: 345      1640 1 !
: 346      1641 1 !
: 347      1642 1 EXTERNAL LITERAL
: 348      1643 1   BAS$K_OUTOF_DAT : UNSIGNED (8),
: 349      1644 1   BAS$K_ENDFILEDEV : UNSIGNED (8),
: 350      1645 1   BAS$K_NOTENODAT : UNSIGNED (8),
: 351      1646 1   BAS$K_TOOLITDAT : UNSIGNED (8),
: 352      1647 1   BAS$K_RECFILETOO : UNSIGNED (8);
: 353      1648 1
: 354      1649 1 !
: 355      1650 1

```

```

! Signal a BASIC error
! Signal fatal Basic error
! Signal a BASIC I/O error
! Signal CTRL/C

```

```

! out of data (READ)
! end of file on device
! not enough data
! ANSI for above
! record in file too long

```

```

: 357      1651 1 GLOBAL ROUTINE BAS$WAIT (
: 358      1652 1     TIME
: 359      1653 1     ) : NOVALUE =
: 360      1654 1
: 361      1655 1 ++
: 362      1656 1 FUNCTIONAL DESCRIPTION:
: 363      1657 1
: 364      1658 1     Limits the time any input I/O statement ( INPUT, INPUT LINE, LINPUT,
: 365      1659 1     MAT 'all above', GET ) to any terminal will wait. If the user does not
: 366      1660 1     reply before the indicated number of seconds an error trap ( which the
: 367      1661 1     user can intercept ) will be taken. WAIT is a module level OWN in this
: 368      1662 1     module.
: 369      1663 1
: 370      1664 1
: 371      1665 1 FORMAL PARAMETERS:
: 372      1666 1
: 373      1667 1     TIME.rl.v     Number of seconds to wait, max.
: 374      1668 1
: 375      1669 1 IMPLICIT INPUTS:
: 376      1670 1
: 377      1671 1     The module level OWN WAIT
: 378      1672 1
: 379      1673 1 IMPLICIT OUTPUTS:
: 380      1674 1
: 381      1675 1     Writes to the module level OWN WAIT the number of seconds given
: 382      1676 1
: 383      1677 1 ROUTINE VALUE:
: 384      1678 1
: 385      1679 1     None
: 386      1680 1
: 387      1681 1 SIDE EFFECTS:
: 388      1682 1
: 389      1683 1     None
: 390      1684 1
: 391      1685 1 --
: 392      1686 1
: 393      1687 2 BEGIN
: 394      1688 2 +
: 395      1689 2     If the WAIT time is unreasonable then force it to the acceptable range.
: 396      1690 2     This is until the correct error message is cooked up for this error.
: 397      1691 2     WAIT is a module level OWN.
: 398      1692 2 -
: 399      1693 2     WAIT = MIN ( ABS(.TIME) , 255 );
: 400      1694 2     RETURN;
: 401      1695 1     END;

```

```
!Limit input wait time
!Seconds to limit time

```

```
!End of BAS$WAIT

```

```

.TITLE BAS$REC_PROC
.IDENT \1-095\

.PSECT _BAS$DATA,NOEXE, PIC,2

00000000 00000 RECOUNT: .LONG 0
0000 00004 WAIT: .WORD 0

.EXTRN BAS$$SIGNAL, BAS$$STOP IO
.EXTRN BAS$$SIGNAL_IO, BAS$$SIGNAL_CTRL C

```



```

                    50      04      AC      D0 00000
                    03      18 00006
000000FF      50      50      CE 00008
                    50      D1 0000B 1$:
                    04      15 00012
00000000'      50      FF      8F 9A 00014
                    50      B0 00018 2$:
                    04      0001F

```

```

.EXTRN BASSK_OUTOF_DAT
.EXTRN BASSK_ENDFILDEV
.EXTRN BASSK_NOTENODAT
.EXTRN BASSK_TOOLITDAT
.EXTRN BASSK_RECFILETOO

```

.PSECT \_BAS\$CODE,NOWRT, SHR, PIC,2

```

.ENTRY BASSWAIT, Save nothing
MOVL   TIME, R0
BGEQ   1$
MNEGL  R0, R0
CML    R0, #255
BLEQ   2$
MOVZBL #255, R0
MOVW   R0, WAIT
RET

```

```

: 1651
: 1693
:
:
:
: 1695

```

; Routine Size: 32 bytes, Routine Base: \_BAS\$CODE + 0000

; 402 1696 1

```
: 404      1697 1 ROUTINE BASS$READ_WAIT      !Read the module level OWN WAIT
: 405      1698 1      : =
: 406      1699 1
: 407      1700 1      ++
: 408      1701 1      FUNCTIONAL DESCRIPTION:
: 409      1702 1
: 410      1703 1      Read the module level OWN WAIT and return it. The value of this
: 411      1704 1      function is the current contents of wait. All routines that need
: 412      1705 1      this value must call this routine.
: 413      1706 1
: 414      1707 1
: 415      1708 1      FORMAL PARAMETERS:
: 416      1709 1
: 417      1710 1      NONE
: 418      1711 1
: 419      1712 1      IMPLICIT INPUTS:
: 420      1713 1
: 421      1714 1      Reads the module level OWN WAIT
: 422      1715 1
: 423      1716 1      IMPLICIT OUTPUTS:
: 424      1717 1
: 425      1718 1      None
: 426      1719 1
: 427      1720 1      ROUTINE VALUE:
: 428      1721 1
: 429      1722 1      The contents of module level OWN WAIT
: 430      1723 1
: 431      1724 1      SIDE EFFECTS:
: 432      1725 1
: 433      1726 1      None
: 434      1727 1      --
: 435      1728 1
: 436      1729 2      BEGIN
: 437      1730 2      +
: 438      1731 2      Just return the value of the module level OWN WAIT
: 439      1732 2      -
: 440      1733 2      RETURN .WAIT;
: 441      1734 1      END;      !End of BASS$READ_WAIT
```

0000 00000 BASS\$READ WAIT:

```
50 00000000' EF 3C 00002      .WORD Save nothing
                   04 00C09      MOVZWL WAIT, R0
                   RET
```

```
: 1697
: 1733
: 1734
```

; Routine Size: 10 bytes, Routine Base: \_BASS\$CODE + 0020

; 442 1735 1



```
: 444      1736 1 GLOBAL ROUTINE BASS$RECOUNT      ! RECOUNT
: 445      1737 1      : =
: 446      1738 1
: 447      1739 1 !++
: 448      1740 1 FUNCTIONAL DESCRIPTION:
: 449      1741 1
: 450      1742 1      This routine supports the Basic RECOUNT function. It returns the number
: 451      1743 1      of bytes read on the last Get. It utilizes a piece of OWN storage which
: 452      1744 1      is written to by the record processing levels which do Gets. In order
: 453      1745 1      to keep the OWN storage from having to be global, this routine is included
: 454      1746 1      in this module.
: 455      1747 1
: 456      1748 1 FORMAL PARAMETERS:
: 457      1749 1
: 458      1750 1      NONE
: 459      1751 1
: 460      1752 1 IMPLICIT INPUTS:
: 461      1753 1
: 462      1754 1      RECOUNT.rl      The number of bytes read on the last GET
: 463      1755 1
: 464      1756 1 IMPLICIT OUTPUTS:
: 465      1757 1
: 466      1758 1      NONE
: 467      1759 1
: 468      1760 1 ROUTINE VALUE:
: 469      1761 1
: 470      1762 1      NUM_OF_BYTES.wl.v      number of bytes read on last Get
: 471      1763 1
: 472      1764 1 SIDE EFFECTS:
: 473      1765 1
: 474      1766 1 --
: 475      1767 1
: 476      1768 2 BEGIN
: 477      1769 2 RETURN .RECOUNT
: 478      1770 1 END;      ! End of BASS$RECOUNT
```

```
50 00000000' EF 0000 00000
                  D0 00002
                  04 00009
```

```
.ENTRY BASS$RECOUNT, Save nothing
MOVL  RECOUNT, R0
RET
```

```
: 1736
: 1769
: 1770
```

; Routine Size: 10 bytes, Routine Base: \_BASS\$CODE + 002A

; 479 1771 1

```
: 481      1772 1 GLOBAL ROUTINE BAS$$RECOU_INIT : NOVALUE =      ! Initialize RECOUNT
: 482      1773 1
: 483      1774 1 !++
: 484      1775 1 FUNCTIONAL DESCRIPTION:
: 485      1776 1
: 486      1777 1
: 487      1778 1      This routine initializes the RECOUNT variable. It is used before a RUN
: 488      1779 1      compiler command in case the previous run of the user's program left
: 489      1780 1      something in RECOUNT.
: 490      1781 1
: 491      1782 1 FORMAL PARAMETERS:
: 492      1783 1
: 493      1784 1      NONE
: 494      1785 1
: 495      1786 1 IMPLICIT INPUTS:
: 496      1787 1
: 497      1788 1      NONE
: 498      1789 1
: 499      1790 1 IMPLICIT OUTPUTS:
: 500      1791 1
: 501      1792 1      RECOUNT.wl      Always set to zero.
: 502      1793 1
: 503      1794 1 ROUTINE VALUE:
: 504      1795 1
: 505      1796 1      NONE
: 506      1797 1
: 507      1798 1 SIDE EFFECTS:
: 508      1799 1
: 509      1800 1 --
: 510      1801 1
: 511      1802 2 BEGIN
: 512      1803 2 RECOUNT = 0;
: 513      1804 1 END;      ! End of BAS$$RECOU_INIT
```

```
00000000' EF 0000 00000
                D4 00002
                04 00008
```

```
.ENTRY BAS$$RECOU_INIT, Save nothing
CLRL RECOUNT
RET
```

```
: 1772
: 1803
: 1804
```

; Routine Size: 9 bytes, Routine Base: \_BAS\$CODE + 0034

; 514 1805 1



```
516 1806 1 GLOBAL ROUTINE BAS$$BLNK_LINE (      ! write a blank line
517 1807 1     FORMAT_CHAR) : CALL_CCB NOVALUE =
518 1808 1
519 1809 1 !++
520 1810 1 FUNCTIONAL DESCRIPTION:
521 1811 1
522 1812 1     Print out a blank line. This is needed between arrays.
523 1813 1
524 1814 1 FORMAL PARAMETERS:
525 1815 1
526 1816 1     FORMAT_CHAR.rlu.v                the format character last used
527 1817 1
528 1818 1 IMPLICIT INPUTS:
529 1819 1
530 1820 1     NONE
531 1821 1
532 1822 1 IMPLICIT OUTPUTS:
533 1823 1
534 1824 1     NONE
535 1825 1
536 1826 1 COMPLETION CODES:
537 1827 1
538 1828 1     NONE
539 1829 1
540 1830 1 SIDE EFFECTS:
541 1831 1
542 1832 1     NONE
543 1833 1
544 1834 1 !--
545 1835 1
546 1836 2 BEGIN
547 1837 2
548 1838 2 EXTERNAL REGISTER
549 1839 2     CCB : REF BLOCK [, BYTE];
550 1840 2
551 1841 2 LOCAL
552 1842 2     RMS_STATUS;
553 1843 2
554 1844 2 !+
555 1845 2 Actually put out the blank line here.
556 1846 2 !-
557 1847 2     CCB [RAB$W_RSZ] = 0;
558 1848 2     CCB [LUB$A_BUF_PTR] = .CCB [LUB$A_BUF_BEG];
559 1849 2     CCB [LUB$B_BAS_VFC1] = BAS$K_LF;
560 1850 2     CCB [LUB$B_BAS_VFC2] = BAS$K_CR;
561 1851 2
562 1852 2     RMS_STATUS = $PUT (RAB = .CCB);
563 1853 2
564 1854 2 IF .RMS_STATUS EQL RMS$_CONTROL_C
565 1855 2 THEN
566 1856 2     BAS$$SIGNAL_CTRL_C ();
567 1857 2
568 1858 2 IF NOT .RMS_STATUS
569 1859 2 THEN
570 1860 2     PUT_ERROR (K_STOP);
571 1861 2
572 1862 2 RETURN;
```

; 573 1863 1 END;

!End of BAS\$\$BLNK\_LINE

				0004 00000	.EXTRN SYSS\$PUT	
		22	AB B4 00002	.ENTRY BAS\$\$BLNK_LINE, Save R2		: 1806
	BO AB	BC	AB D0 00005	CLRW 34(CCB)		: 1847
	DA AB	8D01	8F B0 0000A	MOVL -68(CCB), -80(CCB)		: 1848
			5B DD 00010	MOVW #36097, -38(CCB)		: 1849
00000000G	00		01 FB 00012	PUSHL CCB		: 1852
	52		50 D0 00019	CALLS #1, SYSS\$PUT		
00010651	8F		52 D1 0001C	MOVL R0, RMS_STATUS		
			07 12 00023	CMPL RMS_STATUS, #67153		: 1854
00000000G	00		00 FB 00025	BNEQ 1\$		
	07		52 E8 0002C 1\$:	CALLS #0, BAS\$\$SIGNAL_CTRL		: 1856
			7E D4 0002F	BLBS RMS_STATUS, 2\$		: 1858
0000V	CF		01 FB 00031	CLRL -(SP)		: 1860
			04 00036 2\$:	CALLS #1, PUT_ERROR		: 1863
				RET		

; Routine Size: 55 bytes, Routine Base: \_BAS\$CODE + 003D

; 574 1864 1



```
576 1865 1 GLOBAL ROUTINE BASS$REC_MPR1          ! Write Mat Print record
577 1866 1   : JSB_REC1 NOVALUE =
578 1867 1
579 1868 1 ++
580 1869 1 FUNCTIONAL DESCRIPTION:
581 1870 1
582 1871 1   Write one sequential formatted record and initialize for the next
583 1872 1   BASS$REC_MPR1 writes one record for 10 MAT PRINT A() and then
584 1873 1   initializes the output buffer and returns start and end+1 of user
585 1874 1   part of record buffer to be filled by caller.
586 1875 1   FLR records are space padded.
587 1876 1
588 1877 1 FORMAL PARAMETERS:
589 1878 1
590 1879 1   NONE
591 1880 1
592 1881 1 IMPLICIT INPUTS:
593 1882 1
594 1883 1   LUB$V_FORM_CHAR      =1, comma or semicolon format character
595 1884 1   LUB$W_RBUF_SIZE     Size (bytes) allocated for record buffer at OPEN.
596 1885 1   LUB$A_RBUF_ADR      Address of record buffer from OPEN
597 1886 1   LUB$A_BUF_END       points to last char inserted into buffer
598 1887 1                       by UDF level I/O.
599 1888 1   LUB$V_FORCIBLE       Indicates a forcible device
600 1889 1   LUB$V_OUTBUF_DR    Indicates that there is valid data in the output
601 1890 1                       buffer
602 1891 1   RAB$W_RSZ        Record size
603 1892 1
604 1893 1 IMPLICIT OUTPUTS:
605 1894 1
606 1895 1   LUB$B_BAS_VFC2       'Post' carriage control for terminal devices
607 1896 1   LUB$A_BUF_PTR     Address of next char in user part
608 1897 1                       of record buffer
609 1898 1   LUB$A_BUF_END     Address of last+1 char in user part
610 1899 1                       of record buffer
611 1900 1   LUB$V_OUTBUF_DR   indicates valid data in the output buffer
612 1901 1   LUB$A_BUF_BEG    Beginning of the user buffer
613 1902 1   RAB$L_RBF       Pointer to the user record buffer.
614 1903 1
615 1904 1 ROUTINE VALUE:
616 1905 1
617 1906 1   NONE
618 1907 1
619 1908 1 SIDE EFFECTS:
620 1909 1
621 1910 1   NONE
622 1911 1 --
623 1912 1
624 1913 2 BEGIN
625 1914 2
626 1915 2 EXTERNAL REGISTER
627 1916 2   CCB : REF BLOCK [, BYTE];
628 1917 2
629 1918 2 LOCAL
630 1919 2   RMS_STATUS;
631 1920 2
632 1921 2 !+
```

```

: 633      1922 2      ! If there is no format character, then set the 'pre' and 'post'
: 634      1923 2      ! carriage control to delimit a record.
: 635      1924 2      !-
: 636      1925 2
: 637      1926 2
: 638      1927 2      IF NOT .CCB [LUB$V_FORM_CHAR]
: 639      1928 2      THEN
: 640      1929 2          BEGIN
: 641      1930 2              CCB [LUB$B_BAS_VFC1] = BAS$K_LF;
: 642      1931 2              CCB [LUB$B_BAS_VFC2] = BAS$K_CR;
: 643      1932 2          END;
: 644      1933 2
: 645      1934 2      !+ Set recordsize to actual length of record
: 646      1935 2      !-
: 647      1936 2
: 648      1937 2      CCB [RAB$W_RSZ] = .CCB [LUB$A_BUF_PTR] - .CCB [LUB$A_BUF_BEG];
: 649      1938 2
: 650      1939 2      !+
: 651      1940 2      ! Output buffer to RMS and check for errors
: 652      1941 2      ! If errors, SIGNAL BAS$_FATSYSIO (12='FATAL SYSTEM I/O FAILURE')
: 653      1942 2      !-
: 654      1943 2
: 655      1944 2      CCB [RAB$L_RBF] = .CCB [LUB$A_RBUF_ADR];
: 656      1945 2      CCB [LUB$V_OUTBUF_DR] = 0;
: 657      1946 2
: 658      1947 2      RMS_STATUS = $PUT (RAB = .CCB);
: 659      1948 2
: 660      1949 2      IF .RMS_STATUS EQL RMS$_CONTROL_C
: 661      1950 2      THEN
: 662      1951 2          BAS$$SIGNAL_CTRL_C ();
: 663      1952 2
: 664      1953 2      IF NOT .RMS_STATUS
: 665      1954 2      THEN
: 666      1955 2
: 667      1956 2      !+
: 668      1957 2      ! Not OPEN or CONNECT - RMS record operation
: 669      1958 2      !-
: 670      1959 2
: 671      1960 2          PUT_ERROR (K_STOP);
: 672      1961 2
: 673      1962 2      !+
: 674      1963 2      ! Return next output buffer start and end addresses
: 675      1964 2      !-
: 676      1965 2
: 677      1966 2      CCB [LUB$A_BUF_PTR] = .CCB [LUB$A_RBUF_ADR];
: 678      1967 2      CCB [LUB$A_BUF_END] = .CCB [LUB$A_RBUF_ADR] + .CCB [LUB$W_RBUF_SIZE];
: 679      1968 2      RETURN;
: 680      1969 2      END;
                                     ! End of routine - BAS$$REC_MPR1
```

```

                                52 DD 00000 BAS$$REC_MPR1::
                                PUSHL R2
06 FE AB 8D01 02 E0 00002 BBS #2, -2(CCB), 1$
                                DA AB 8F B0 00007 MOVW #36097, -38(CCB)
```

```

: 1865
: 1926
: 1929
```



BAS\$\$REC\_PROC  
1-095

J 5  
16-Sep-1984 01:01:12  
14-Sep-1984 11:56:35

VAX-11 Bliss-32 V4.0-742  
[BASRTL.SRC]BASRECPRO.B32;1

Page 17  
(8)

22	AB	B0	AB	BC	AB	A3	0000D	1\$:	SUBW3	-68(CCB), -80(CCB), 34(CCB)	:	1937
		28	AB	EC	AB	D0	00014		MOVL	-20(CCB), 40(CCB)	:	1944
		FE	AB		08	8A	00019		BICB2	#8, -2(CCB)	:	1945
					5B	DD	0001D		PUSHL	CCB	:	1947
		00000000G	00		01	FB	0001F		CALLS	#1, SYSSPUT	:	
			52		50	D0	00026		MOVL	R0, RMS STATUS	:	
		00010651	8F		52	D1	00029		CMPL	RMS STATUS, #67153	:	1949
					07	12	00030		BNEQ	2\$	:	
		00000000G	00		00	FB	00032		CALLS	#0, BAS\$\$SIGNAL_CTRL	:	1951
			07		52	E8	00039	2\$:	BLBS	RMS STATUS, 3\$	:	1953
					7E	D4	0003C		CLRL	-(SP)	:	1960
		0000V	CF		01	FB	0003E		CALLS	#1, PUT ERROR	:	
		B0	AB	EC	AB	D0	00043	3\$:	MOVL	-20(CCB), -80(CCB)	:	1966
			50	D2	AB	3C	00048		MOVZWL	-46(CCB), R0	:	1967
		B4	AB	EC	BB40	9E	0004C		MOVAB	@-20(CCB)[R0], -76(CCB)	:	
					04	BA	00052		POPR	#^M<R2>	:	1969
						05	00054		RSB		:	

; Routine Size: 85 bytes, Routine Base: \_BAS\$CODE + 0074

; 681 1970 1

```
683 1971 1 GLOBAL ROUTINE BASS$REC_MPR9          ! Mat Write sequential
684 1972 1 : JSB_REC9 NOVALUE =
685 1973 1
686 1974 1 !++
687 1975 1 FUNCTIONAL DESCRIPTION:
688 1976 1
689 1977 1 This routine does not write a record. Presumably the MAT PRINT element
690 1978 1 transmitter took care of all of that. Since we do not want a blank line
691 1979 1 after the array, there is no need to write anything here.
692 1980 1
693 1981 1 FORMAL PARAMETERS:
694 1982 1
695 1983 1 NONE
696 1984 1
697 1985 1 IMPLICIT INPUTS:
698 1986 1
699 1987 1 LUB$W_RBUF_SIZE      Size (bytes) allocated for record buffer at OPEN.
700 1988 1 LUB$A_RBUF_ADR        Address of record buffer from OPEN
701 1989 1
702 1990 1 IMPLICIT OUTPUTS:
703 1991 1
704 1992 1 LUB$A_BUF_PTR        Address of next char in user part
705 1993 1                   of record buffer
706 1994 1 LUB$A_BUF_END      Address of last+1 char in user part
707 1995 1                   of record buffer
708 1996 1
709 1997 1 ROUTINE VALUE:
710 1998 1
711 1999 1 NONE
712 2000 1
713 2001 1 SIDE EFFECTS:
714 2002 1
715 2003 1 NONE
716 2004 1 --
717 2005 1
718 2006 2 BEGIN
719 2007 2
720 2008 2 EXTERNAL REGISTER
721 2009 2 CCB : REF BLOCK [, BYTE];
722 2010 2
723 2011 2 !+
724 2012 2 ! Return next output buffer start and end addresses
725 2013 2 !-
726 2014 2
727 2015 2 CCB [LUB$A_BUF_PTR] = .CCB [LUB$A_RBUF_ADR];
728 2016 2 CCB [LUB$A_BUF_END] = .CCB [LUB$A_RBUF_ADR] + .CCB [LUB$W_RBUF_SIZE];
729 2017 2 RETURN;
730 2018 1 END;          ! END BASS$REC_MPR19
```

```
B0 AB EC AB D0 0000 BASS$REC_MPR9::
B4 AB D2 AB 3C 0005 MOVL -20(CCB), -80(CCB)
EC BB40 9E 00009 MOVZWL -46(CCB), R0
MOVAB @-20(CCB)[R0], -76(CCB)
```

```
: 2015
: 2016
:
```



BASS\$REC\_PROC  
1-095

L 5  
16-Sep-1984 01:01:12  
14-Sep-1984 11:56:35

VAX-11 Bliss-32 V4.0-742  
[BASRTL.SRC]BASRECPRO.B32;1

Page 19  
(9)

05 0000F

RSB

; 2018

; Routine Size: 16 bytes, Routine Base: \_BASS\$CODE + 00C9

; 731 2019 1

```
733 2020 1 GLOBAL ROUTINE BASS$REC_WSL9          ! Write sequential formatted
734 2021 1   : JSB_REC9 NOVALUE =
735 2022 1
736 2023 1
737 2024 1 ++
738 2025 1 FUNCTIONAL DESCRIPTION:
739 2026 1   Write one sequential formatted record and initialize for the next
740 2027 1   BASS$REC_WSF1 (and BASS$REC_WSL9) writes one output buffer and then
741 2028 1   initializes the output buffer and returns start and end+1 of user
742 2029 1   part of record buffer to be filled by caller.
743 2030 1   FLR records are space padded.
744 2031 1   /logical record number is incremented/.
745 2032 1
746 2033 1 FORMAL PARAMETERS:
747 2034 1
748 2035 1   NONE
749 2036 1
750 2037 1 IMPLICIT INPUTS:
751 2038 1
752 2039 1   ISB$V_PRINT_INI      flag to indicate whether there was an element
753 2040 1                       transmitter
754 2041 1   LUB$W_RBUF_SIZE     Size (bytes) allocated for record buffer at OPEN.
755 2042 1   LUB$A_RBUF_ADR     Address of record buffer from OPEN
756 2043 1   LUB$A_BUF_END     points to last char inserted into buffer
757 2044 1                       by UDF level I/O.
758 2045 1   LUB$V_FORM_CHAR    The last element transmitter ended in a comma
759 2046 1                       or semicolon format char.
760 2047 1   LUB$V_FORCIBLE     Indicates a forcible device
761 2048 1   LUB$V_OUTBUF_DR    Indicates that there is valid data in the output
762 2049 1                       buffer
763 2050 1   RAB$W_RSZ         Record size
764 2051 1
765 2052 1 IMPLICIT OUTPUTS:
766 2053 1
767 2054 1   ISB$V_PRINT_INI      reset flag
768 2055 1   LUB$B_BAS_VFC2     'Post' carriage control for terminal devices
769 2056 1   LUB$A_BUF_PTR     Address of next char in user part
770 2057 1                       of record buffer
771 2058 1   LUB$A_BUF_END     Address of last+1 char in user part
772 2059 1                       of record buffer
773 2060 1   LUB$V_OUTBUF_DR    indicates valid data in the output buffer
774 2061 1   LUB$A_BUF_BEG     Beginning of the user buffer
775 2062 1   RAB$L_RBF        Pointer to the user record buffer.
776 2063 1
777 2064 1 ROUTINE VALUE:
778 2065 1
779 2066 1   NONE
780 2067 1
781 2068 1 SIDE EFFECTS:
782 2069 1
783 2070 1   NONE
784 2071 1 --
785 2072 1
786 2073 2 BEGIN
787 2074 2
788 2075 2 EXTERNAL REGISTER
789 2076 2   CCB : REF BLOCK [, BYTE];
```



```

790      2077 2
791      2078
792      2079
793      2080
794      2081
795      2082
796      2083
797      2084
798      2085
799      2086
800      2087
801      2088
802      2089
803      2090
804      2091
805      2092
806      2093
807      2094
808      2095
809      2096
810      2097
811      2098
812      2099
813      2100
814      2101
815      2102
816      2103
817      2104
818      2105
819      2106
820      2107
821      2108
822      2109
823      2110
824      2111
825      2112
826      2113
827      2114
828      2115
829      2116
830      2117
831      2118
832      2119
833      2120
834      2121
835      2122
836      2123
837      2124
838      2125
839      2126
840      2127
841      2128
842      2129
843      2130
844      2131
845      2132
846      2133 2

LOCAL
  RMS_STATUS;

  !+
  ! If last element ended with a format character and not a terminal device
  ! then return to caller without writing anything. With CR format, we must
  ! PUT a whole record.
  !-

  IF .CCB [LUB$V_FORM_CHAR] AND NOT .CCB [LUB$V_FORCIBLE] THEN RETURN;

  !+
  ! Set the 'post' carriage control to carriage return
  ! if the last element transmitter had no format character following.
  !-

  IF NOT .CCB [LUB$V_FORM_CHAR] THEN CCB [LUB$B_BAS_VFC2] = BAS$K_CR;

  !+
  ! Set recordsize to actual length of record
  !-

  CCB [RAB$W_RSZ] = .CCB [LUB$A_BUF_PTR] - .CCB [LUB$A_BUF_BEG];

  !+
  ! Output buffer to RMS and check for errors
  ! If errors, SIGNAL BAS$_FATSYSIO (12='FATAL SYSTEM I/O FAILURE')
  !-

  CCB [RAB$L_RBF] = .CCB [LUB$A_BUF_BEG];

  !+
  ! Write something if there is something in the buffer or if there was no
  ! element transmitter.
  !-

  IF .CCB [LUB$V_OUTBUF_DR] OR .CCB [ISB$V_PRINT_INI]
  THEN
    RMS_STATUS = $PUT (RAB = .CCB);

    IF .RMS_STATUS EQL RMS$_CTRLC
    THEN
      BAS$$SIGNAL_CTRLC ();

    IF NOT .RMS_STATUS
    THEN
      PUT_ERROR (K_STOP);

    CCB [LUB$V_OUTBUF_DR] = 0;
    CCB [ISB$V_PRINT_INI] = 0;

    !+
    ! If there is no format character then set the 'pre' carriage control to LF
    ! for the next record. This is recursive I/O and the rest of the list when
    ! we return should be written on the next line.
    !-

```



```
! END OF ROUTINE
```

: 857 2144 1



```
859 2145 1 GLOBAL ROUTINE BASS$REC_RSLO ! Read initialization
860 2146 1 : JSB_RECO NOVALUE =
861 2147 1
862 2148 1 ++
863 2149 1 FUNCTIONAL DESCRIPTION:
864 2150 1
865 2151 1 BASS$REC_RSFO (and BASS$REC_RSFI) reads one record if this is not a terminal.
866 2152 1 Then return start and end+1 of user
867 2153 1 part of record to be processed as input.
868 2154 1
869 2155 1 FORMAL PARAMETERS:
870 2156 1
871 2157 1 NONE
872 2158 1
873 2159 1 IMPLICIT INPUTS:
874 2160 1
875 2161 1 LUB$W_RBUF_SIZE Size of record buffer allocated in OPEN.
876 2162 1 LUB$A_RBUF_ADR Address of record buffer from OPEN.
877 2163 1 LUB$V_TERM_DEV flag in LUB which indicates a terminal device.
878 2164 1 RAB$W_RSZ word in the RAB which contains the buffer size.
879 2165 1 RAB$L_RBF longword in RAB which points to the buffer.
880 2166 1 LUB$L_WAIT_TIME Max time to wait for input, in seconds.
881 2167 1 WAIT The module level OWN WAIT
882 2168 1
883 2169 1 IMPLICIT OUTPUTS:
884 2170 1
885 2171 1 RECOUNT Global storage to hold number of bytes read from
886 2172 1 last input.
887 2173 1 LUB$L_LOG_RECNO Increment logical record number
888 2174 1 of next record to be read.
889 2175 1 LUB$A_BUF_PTR points to first char of user part of
890 2176 1 record buffer.
891 2177 1 LUB$A_BUF_END points to end+1 of user part of
892 2178 1 record buffer.
893 2179 1
894 2180 1 ROUTINE VALUE:
895 2181 1
896 2182 1 NONE
897 2183 1
898 2184 1 SIDE EFFECTS:
899 2185 1
900 2186 1 Reads next record from file on this logical unit.
901 2187 1 Throws away things that are pending in the Print buffer for non-terminal
902 2188 1 devices.
903 2189 1 SIGNALS RMS errors directly.
904 2190 1 SIGNALS BASS$K_TIMLIMEXC if a wait time was specified and we
905 2191 1 exceed it.
906 2192 1 --
907 2193 1
908 2194 2 BEGIN
909 2195 2
910 2196 2 EXTERNAL REGISTER
911 2197 2 (CB : REF BLOCK [, BYTE]);
912 2198 2
913 2199 2 LITERAL
914 2200 2 K_ESCAPE = %X'1B',
915 2201 2 K_CR = %X'0D';
```



```

916 2202 2
917 2203
918 2204 LOCAL
919 2205 RMS_STATUS,
920 2206 WAIT_TIME; ! Current wait time
921 2207
922 2208 + If a timeout has been specified, store information in the RAB to tell
923 2209 RMS about it. If no timeout has been specified, clear the TMO bit
924 2210 in case there was an earlier timeout specified.
925 2211
926 2212 +
927 2213 If WAIT is zero then use the LUB's wait. This is to provide upward compatibility
928 2214 , i.e. existing EXE's can run with the LUB wait value in V2.2.
929 2215
930 2216 WAIT_TIME = ( IF ( .WAIT EQL 0 ) THEN .CCB [ LUB$WAIT_TIME ] ELSE .WAIT );
931 2217
932 2218 IF ( .WAIT_TIME EQL 0 )
933 2219 THEN
934 2220 CCB [ RAB$V_TMO ] = 0
935 2221 ELSE
936 2222 BEGIN
937 2223 CCB [ RAB$B_TMO ] = .WAIT_TIME;
938 2224 CCB [ RAB$V_TMO ] = 1;
939 2225 END;
940 2226
941 2227
942 2228 +
943 2229 Set the Read-no-echo RMS bit based on the user's last call to
944 2230 ECHO or NOECHO.
945 2231
946 2232 CCB [ RAB$V_RNE ] = .CCB [ LUB$V_NOECHO ];
947 2233
948 2234 +
949 2235 Check to see if this is a terminal device. If this is NOT
950 2236 a terminal then do a GET. GETs for terminals are done each time more
951 2237 data are needed.
952 2238 Read record into buffer using RMS and check for errors
953 2239
954 2240
955 2241 IF ( NOT .CCB [ LUB$V_TERM_DEV ] OR .CCB [ LUB$V_ANSI ] )
956 2242 THEN
957 2243 BEGIN
958 2244
959 2245 LOCAL
960 2246 TEMP_CCB : REF BLOCK [, BYTE]; ! Temporary CCB
961 2247 TEMP_CCB = .CCB [ LUB$A_BUDDY_PTR ];
962 2248
963 2249 +
964 2250 If there is something pending in the Print buffer, then $PUT it.
965 2251 It cannot become a prompt, because RMS will throw away prompts
966 2252 to disk files; therefore we must $PUT it.
967 2253
968 2254 IF ( NOT .CCB [ LUB$V_TERM_DEV ] ) AND .TEMP_CCB [ LUB$V_OUTBUF_DR ]
969 2255 THEN
970 2256 BEGIN
971 2257 TEMP_CCB [ RAB$W_RSZ ] = .TEMP_CCB [ LUB$A_BUF_PTR ] - .TEMP_CCB [ LUB$A_BUF_BEG ];
972 2258 TEMP_CCB [ RAB$L_RBF ] = .TEMP_CCB [ LUB$A_BUF_BEG ];

```



```

: 973      2259 4
: 974      2260 4
: 975      2261 4
: 976      2262 4
: 977      2263 4
: 978      2264 4
: 979      2265 4
: 980      2266 4
: 981      2267 4
: 982      2268 4
: 983      2269 4
: 984      2270 4
: 985      2271 4
: 986      2272 4
: 987      2273 4
: 988      2274 4
: 989      2275 4
: 990      2276 4
: 991      2277 4
: 992      2278 4
: 993      2279 4
: 994      2280 4
: 995      2281 4
: 996      2282 4
: 997      2283 4
: 998      2284 4
: 999      2285 4
1000      2286 4
1001      2287 4
1002      2288 4
1003      2289 4
1004      2290 4
1005      2291 4
1006      2292 4
1007      2293 4
1008      2294 4
1009      2295 4
1010      2296 4
1011      2297 4
1012      2298 4
1013      2299 4
1014      2300 4
1015      2301 4
1016      2302 4
1017      2303 4
1018      2304 4
1019      2305 4
1020      2306 4
1021      2307 4
1022      2308 4
1023      2309 4
1024      2310 4
1025      2311 4
1026      2312 4
1027      2313 4
1028      2314 4
: 1029      2315 4

      RMS_STATUS = $PUT (RAB = .TEMP_CCB);
      IF .RMS_STATUS EQL RMS$_CONTROL_C
      THEN
        BAS$$SIGNAL_CTRL_C ();
      IF NOT .RMS_STATUS
      THEN
        PUT_ERROR (K_STOP);
      END;

      TEMP_CCB [LUB$_BUF_PTR] = .TEMP_CCB [LUB$_BUF_BEG];
      RMS_STATUS = $GET (RAB = .CCB);
      IF .RMS_STATUS EQL RMS$_CONTROL_C
      THEN
        BAS$$SIGNAL_CTRL_C ();
      IF NOT .RMS_STATUS
      THEN
        GET_ERROR (K_STOP);

      +
      | Set RECOUNT to the number of bytes read
      | If the file is a terminal format file, then RECOUNT has to be
      | adjusted for the carriage control terminator. Because RMS does not return
      | a terminator for a file, we unconditionally put a CRLF on the end and
      | bump RECOUNT by 2.
      -
      RECOUNT = .CCB [RAB$_RSZ] + (IF (.CCB [LUB$_TERM_FOR]) AND ((.CCB [LUB$_RAT] AND FAB$_CR) NEQU 0
      THEN 2 ELSE 0);

      +
      | Put the CR into the STV field since RMS doesn't
      | We should only do this if the record attributes indicate a CR format.
      -
      IF (.CCB [LUB$_RAT] AND FAB$_CR) NEQU 0 THEN CCB [RAB$_STV] = 13;

      +
      | Return start-1 and end+1 address of record just read
      -
      CCB [LUB$_BUF_PTR] = .CCB [RAB$_RBF] - 1;
      CCB [LUB$_BUF_END] = .CCB [RAB$_RBF] + .CCB [RAB$_RSZ];
      END
    ELSE
      +
      | This is a terminal. Force a no data in the buffer condition
      | so the first GET is done on the element transmitter after the
      | Prompt (if any) is known.
      -
      BEGIN
```

```
: 1030      2316 3      CCB [LUBSA_BUF_PTR] = .CCB [RABSL_RBF];  
: 1031      2317 3      CCB [LUBSA_BUF_END] = .CCB [LUBSA_BUF_PTR];  
: 1032      2318 2      END;  
: 1033      2319 2  
: 1034      2320 2      RETURN;  
: 1035      2321 1      END;
```

! End of BAS\$\$REC\_RSLO

.EXTRN SYSSGET

```
0C BB 00000 BAS$$REC_RSLO::  
50 00000000' EF 3C 00002 PUSHR #^M<R2,R3> : 2145  
0C 12 00009 MOVZWL WAIT, R0 : 2216  
50 CC AB D0 0000B BNEQ 1$  
06 12 0000F MOVL -52(CCB), WAIT_TIME : 2218  
07 AB 02 8A 00011 BNEQ 1$ : 2220  
08 11 00015 BICB2 #2, 7(CCB)  
50 90 00017 1$: BRB 2$ : 2223  
07 AB 02 88 0001B MOVB WAIT_TIME, 31(CCB)  
00 A0 AB F0 0001F 2$: BISB2 #2, 7(CCB) : 2224  
08 FE AB 05 E1 00026 INSV -96(CCB), #0, #1, 7(CCB) : 2232  
03 A1 AB 04 E0 0002B BBC #5, -2(CCB), 3$ : 2241  
00A3 31 00030 BBS #4, -95(CCB), 3$  
52 BB AB D0 00033 3$: BRW 11$ : 2247  
37 FE AB 05 E0 00037 MOVL -72(CCB), TEMP_CCB : 2254  
32 FE A2 03 E1 0003C BBS #5, -2(CCB), 5$  
22 A2 B0 A2 BC A2 A3 00041 BBC #3, -2(TEMP_CCB), 5$ : 2257  
28 A2 BC A2 D0 00048 SUBW3 -68(TEMP_CCB), -80(TEMP_CCB), 34(TEMP_CCB)  
52 DD 0004D MOVL -68(TEMP_CCB), 40(TEMP_CCB) : 2258  
00 01 FB 0004F PUSHL TEMP_CCB : 2260  
53 50 D0 00056 CALLS #1, SYSSPUT  
00010651 8F 53 D1 00059 MOVL R0, RMS_STATUS  
07 12 00060 CMPL RMS_STATUS, #67153 : 2262  
00 00 FB 00062 BNEQ 4$ : 2264  
07 53 E8 00069 4$: CALLS #0, BAS$$SIGNAL_CTRL C  
7E D4 0006C BLBS RMS_STATUS, 5$ : 2266  
01 FB 0006E CLRL -(SP) : 2268  
BC A2 D0 00073 5$: CALLS #1, PUT_ERROR  
5B DD 00078 MOVL -68(TEMP_CCB), -80(TEMP_CCB) : 2271  
00 01 FB 0007A PUSHL CCB : 2273  
53 50 D0 00081 CALLS #1, SYSSGET  
00010651 8F 53 D1 00084 MOVL R0, RMS_STATUS  
07 12 0008B CMPL RMS_STATUS, #67153 : 2275  
00 00 FB 0008D BNEQ 6$ : 2277  
07 53 E8 00094 6$: CALLS #0, BAS$$SIGNAL_CTRL C  
7E D4 00097 BLBS RMS_STATUS, 7$ : 2279  
01 FB 00099 CLRL -(SP) : 2281  
0A 05 FE AB 04 E1 0009E 7$: CALLS #1, GET_ERROR : 2291  
F6 AB 01 E1 000A3 BBC #4, -2(CCB), 8$  
50 02 D0 000A8 BBC #1, -10(CCB), 8$  
02 11 000AB MOVL #2, R0  
50 D4 000AD BRB 9$  
00000000' EF 51 22 AB 3C 000AF 8$: CLRL R0 : 2298  
04 F6 AB 01 E1 000BB 9$: MOVZWL 34(CCB), R1  
ADDL3 R1, R0, RECOUNT  
BBC #1, -10(CCB), 10$
```



BASS\$REC\_PROC  
1-095

G 6  
16-Sep-1984 01:01:12  
14-Sep-1984 11:56:35

VAX-11 Bliss-32 V4.0-742  
[BASRTL.SRC]BASRECPRO.B32;1

Page 27  
(11)

B0	AB	0C	AB	0D	D0	000C0	MOVL	#13, 12(CCB)	:	
		28	AB	01	C3	000C4	SUBL3	#1, 40(CCB), -80(CCB)	:	2304
			50	22	AB	3C	MOVZWL	34(CCB), R0	:	2305
		B4	AB	28	BB40	9E	MOVAB	@40(CCB)[R0], -76(CCB)	:	
					0A	11	BRB	12\$	:	2241
		B0	AB	28	AB	D0	MOVL	40(CCB), -80(CCB)	:	2316
		B4	AB	B0	AB	D0	MOVL	-80(CCB), -76(CCB)	:	2317
					0C	BA	POPR	#^M<R2,R3>	:	2321
					05	000E2	RSB		:	

; Routine Size: 227 bytes, Routine Base: \_BASS\$CODE + 014E

; 1036 2322 1

```
1038 2323 1 GLOBAL ROUTINE BASS$REC_MINO ! MAT Input initialization
1039 2324 1 : JSB_RECO NOVALUE =
1040 2325 1
1041 2326 1 ++
1042 2327 1 FUNCTIONAL DESCRIPTION:
1043 2328 1
1044 2329 1 BASS$REC_RSFO (and BASS$REC_RSFI) reads one record if this is not a terminal.
1045 2330 1 Then return start and end+1 of user
1046 2331 1 part of record to be processed as input.
1047 2332 1
1048 2333 1 FORMAL PARAMETERS:
1049 2334 1
1050 2335 1 NONE
1051 2336 1
1052 2337 1 IMPLICIT INPUTS:
1053 2338 1
1054 2339 1 LUB$W_RBUF_SIZE Size of record buffer allocated in OPEN.
1055 2340 1 LUB$A_RBUF_ADR Address of record buffer from OPEN.
1056 2341 1 LUB$V_TERM_DEV flag in LUB which indicates a terminal device.
1057 2342 1 RAB$W_RSZ word in the RAB which contains the buffer size.
1058 2343 1 RAB$L_RBF longword in RAB which points to the buffer.
1059 2344 1 LUB$L_WAIT_TIME Max time to wait for input, in seconds.
1060 2345 1 WAIT Module level OWN WAIT
1061 2346 1
1062 2347 1 IMPLICIT OUTPUTS:
1063 2348 1
1064 2349 1 RECOUNT Global storage to hold number of bytes read from
1065 2350 1 last input.
1066 2351 1 LUB$L_LOG_RECNO Increment logical record number
1067 2352 1 of next record to be read.
1068 2353 1 LUB$A_BUF_PTR points to first char of user part of
1069 2354 1 record buffer.
1070 2355 1 LUB$A_BUF_END points to end+1 of user part of
1071 2356 1 record buffer.
1072 2357 1
1073 2358 1 ROUTINE VALUE:
1074 2359 1
1075 2360 1 NONE
1076 2361 1
1077 2362 1 SIDE EFFECTS:
1078 2363 1
1079 2364 1 Reads next record from file on this logical unit.
1080 2365 1 Throws away things that are pending in the Print buffer for non-terminal
1081 2366 1 devices.
1082 2367 1 SIGNALS BASS$FATSYSIO (12='FATAL SYSTEM I/O FAILURE')
1083 2368 1 SIGNALS BASS$ENDFILDEV (11='END-OF-FILE ON DEVICE')
1084 2369 1 SIGNALS BASS$RECFILTOO if record too big
1085 2370 1 SIGNALS BASS$TIMLIMEXC if a wait time was specified and we
1086 2371 1 exceed it.
1087 2372 1 --
1088 2373 1
1089 2374 2 BEGIN
1090 2375 2
1091 2376 2 EXTERNAL REGISTER
1092 2377 2 CCB : REF BLOCK [, BYTE];
1093 2378 2
1094 2379 2 LITERAL
```



```
1095 2380      K_ESCAPE = %X'1B',
1096 2381      K_CR = %X'0D';
1097 2382
1098 2383      LOCAL
1099 2384          RMS_STATUS,
1100 2385          WAIT_TIME;
1101 2386          !Current wait time
1102 2387      +
1103 2388      If a timeout has been specified, store information in the RAB to tell
1104 2389      RMS about it. If no timeout has been specified, clear the TMO bit
1105 2390      in case there was an earlier timeout specified.
1106 2391      -
1107 2392
1108 2393      +
1109 2394      If WAIT is zero then use the LUB's wait. This is to provide upward compatibility
1110 2395      , i.e. existing EXE's can run with the LUB wait value in V2.2.
1111 2396      -
1112 2397      WAIT_TIME = ( IF ( .WAIT EQL 0 ) THEN .CCB [ LUB$L_WAIT_TIME ] ELSE .WAIT );
1113 2398
1114 2399      IF ( .WAIT_TIME EQL 0 )
1115 2400      THEN
1116 2401          CCB [ RAB$V_TMO ] = 0
1117 2402      ELSE
1118 2403          BEGIN
1119 2404              CCB [ RAB$B_TMO ] = .WAIT_TIME;
1120 2405              CCB [ RAB$V_TMO ] = 1;
1121 2406          END;
1122 2407
1123 2408
1124 2409      +
1125 2410      Set the Read-no-echo RMS bit based on the user's last call to
1126 2411      ECHO or NOECHO.
1127 2412      -
1128 2413      CCB [ RAB$V_RNE ] = .CCB [ LUB$V_NOECHO ];
1129 2414
1130 2415      +
1131 2416      Check to see if this is a terminal device. If this is NOT
1132 2417      a terminal then do a GET. GETs for terminals are done each time more
1133 2418      data are needed.
1134 2419      Read record into buffer using RMS and check for errors
1135 2420      If end-of-file, SIGNAL BAS$K_ENDFILDEV (11='END-OF-FILE ON DEVICE')
1136 2421      If record too big for record buffer, SIGNAL BAS$K_RECFLT00.
1137 2422      If errors, SIGNAL BAS$K_FATSYSIO (12='FATAL SYSTEM I/O ERROR')
1138 2423      -
1139 2424
1140 2425      IF ( NOT .CCB [ LUB$V_TERM_DEV ] )
1141 2426      THEN
1142 2427          BEGIN
1143 2428
1144 2429          LOCAL
1145 2430              TEMP_CCB : REF BLOCK [, BYTE];          ! Temporary CCB
1146 2431              TEMP_CCB = .CCB [ LUB$A_BUDDY_PTR ];
1147 2432
1148 2433      +
1149 2434      If there is something pending in the Print buffer, then $PUT it.
1150 2435      It cannot become a prompt, because RMS will throw away prompts
1151 2436      to disk files; therefore we must $PUT it.
```

```
1152 2437 3
1153 2438 3
1154 2439 3
1155 2440 4
1156 2441 4
1157 2442 4
1158 2443 4
1159 2444 4
1160 2445 4
1161 2446 4
1162 2447 4
1163 2448 4
1164 2449 4
1165 2450 4
1166 2451 4
1167 2452 4
1168 2453 4
1169 2454 4
1170 2455 4
1171 2456 4
1172 2457 4
1173 2458 4
1174 2459 4
1175 2460 4
1176 2461 4
1177 2462 4
1178 2463 4
1179 2464 4
1180 2465 4
1181 2466 4
1182 2467 4
1183 2468 4
1184 2469 4
1185 2470 4
1186 2471 4
1187 2472 4
1188 2473 4
1189 2474 4
1190 2475 4
1191 2476 4
1192 2477 4
1193 2478 4
1194 2479 4
1195 2480 4
1196 2481 4
1197 2482 4
1198 2483 4
1199 2484 4
1200 2485 4
1201 2486 4
1202 2487 4
1203 2488 4
1204 2489 4
1205 2490 4
1206 2491 4
1207 2492 4
1208 2493 3

!-
IF (NOT .CCB [LUB$V_TERM_DEV]) AND .TEMP_CCB [LUB$V_OUTBUF_DR]
THEN
  BEGIN
    TEMP_CCB [RAB$W_RSZ] = .TEMP_CCB [LUB$A_BUF_PTR] - .TEMP_CCB [LUB$A_BUF_BEG];
    TEMP_CCB [RAB$L_RBF] = .TEMP_CCB [LUB$A_BUF_BEG];

    RMS_STATUS = $PUT (RAB = .CCB);

    IF .RMS_STATUS EQL RMS$_CONTROL_C
    THEN
      BAS$$SIGNAL_CTRL_C ();

    IF NOT .RMS_STATUS
    THEN
      PUT_ERROR (K_STOP);
    END;

    TEMP_CCB [LUB$A_BUF_PTR] = .TEMP_CCB [LUB$A_BUF_BEG];
    RMS_STATUS = $GET (RAB = .CCB);

    IF .RMS_STATUS EQL RMS$_CONTROL_C
    THEN
      BAS$$SIGNAL_CTRL_C ();

    IF NOT .RMS_STATUS
    THEN
      GET_ERROR (K_STOP);

    !+
    ! Set RECOUNT to the number of bytes read
    ! If the file is a terminal format file, then RECOUNT has to be
    ! adjusted for the carriage control terminator.
    !-
    RECOUNT = .CCB [RAB$W_RSZ] + (IF .CCB [LUB$V_TERM_FOR] THEN SELECT ONEU .CCB [RAB$W_STV0] OF
      SET
      [K_ESCAPE] : .CCB [RAB$W_STV2];
      [K_CR] : 2;
      [OTHERWISE] : 0;
      TES ELSE 0);

    !+
    ! Return start-1 and end+1 address of record just read
    !-
    CCB [LUB$A_BUF_PTR] = .CCB [RAB$L_RBF] - 1;
    CCB [LUB$A_BUF_END] = .CCB [RAB$L_RBF] + .CCB [RAB$W_RSZ];

    !+
    ! Check for an '&' as the last character of the record. If it is there,
    ! it is a continuation character and signifies that there is more data to
    ! come in the next record.
    !-
    IF .(.CCB [LUB$A_BUF_END] - 1) < 0, 8> EQLU K_MAT_CONT_CHAR
```



```
! End of BAS$$REC_MINO
```

PC	BB	00000	BAS\$\$REC	MINO::		
50	00000000	EF	3C	00002	PUSHR	#*M<R2,R3>
		0C	12	00009	MOVZWL	WAIT, R0
50	CC	AB	D0	0000B	BNEQ	1\$
		06	12	0000F	MOVL	-52(CCB), WAIT_TIME
07	AB	02	8A	00011	BNEQ	1\$
		08	11	00015	BICB2	#2, 7(CCB)
1F	AB	50	90	00017	BRB	2\$
07	AB	02	88	0001B	MOVB	WAIT_TIME, 31(CCB)
		00	AB	F0	BISB2	#2, 7(CCB)
01	03	FE	AB	A0	INSV	-96(CCB), #0, #1, 7(CCB)
		52	B8	00BC	BBC	#5, -2(CCB), 3\$
37	FE	AB	05	E1	BRW	12\$
32	FE	A2	03	E1	MOVL	-72(CCB), TEMP_CCB
22	A2	B0	A2	BC	BBS	#5, -2(CCB), 5\$
		28	A2	BC	BBC	#3, -2(TEMP_CCB), 5\$
	00000000G	00	01	FB	SUBW3	-68(TEMP_CCB), -80(TEMP_CCB), 34(TEMP_CCB)
		53	50	D0	MOVL	-68(TEMP_CCB), 40(TEMP_CCB)
	00010651	8F	53	D1	PUSHL	CCB
			07	12	CALLS	#1, SYSSPUT
	00000000G	00	00	FB	MOVL	R0, RMS_STATUS
		07	53	E8	CMPL	RMS_STATUS, #67153
			7E	D4	BNEQ	4\$
					CALLS	#0, BAS\$\$SIGNAL_CTRL
					BLBS	RMS_STATUS, 5\$
					CLRL	-(SP)

	0000V	CF		01	FB	00069		CALLS	#1, PUT_ERROR		
	B0	A2	BC	A2	D0	0006E	5\$:	MOVL	-68(TEMP_CCB), -80(TEMP_CCB)		2455
				5B	DD	00073		PUSHL	CCB		2457
	00000000G	00		01	FB	00075		CALLS	#1, SYSS\$GET		
		53		50	D0	0007C		MOVL	R0, RMS_STATUS		
	00010651	8F		53	D1	0007F		CMPL	RMS_STATUS, #67153		2459
				07	12	00086		BNEQ	6\$		
	00000000G	00		00	FB	00088		CALLS	#0, BASS\$SIGNAL_CTRL		2461
		07		53	E8	0008F	6\$:	BLBS	RMS_STATUS, 7\$		2463
				7E	D4	00092		CLRL	-(SP)		2465
	0000V	CF		01	FB	00094		CALLS	#1, GET_ERROR		
19	FE	AB		04	E1	00099	7\$:	BBC	#4, -2(CCB), 9\$		2473
		50	0C	AB	3C	0009E		MOVZWL	12(CCB), R0		
		1B		50	B1	000A2		CMPL	R0, #27		2475
				06	12	000A5		BNEQ	8\$		
		50	0E	AB	3C	000A7		MOVZWL	14(CCB), R0		
				0C	11	000AB		BRB	10\$		
		0D		50	B1	000AD	8\$:	CMPL	R0, #13		2476
				05	12	000B0		BNEQ	9\$		
		50		02	D0	000B2		MOVL	#2, R0		
				02	11	000B5		BRB	10\$		
				50	D4	000B7	9\$:	CLRL	R0		2473
		51	22	AB	3C	000B9	10\$:	MOVZWL	34(CCB), R1		
00000000'	EF	50		51	C1	000BD		ADDL3	R1, R0, RECOUNT		
B0	AB	28		01	C3	000C5		SUBL3	#1, 40(CCB), -80(CCB)		2484
		50	22	AB	3C	000CB		MOVZWL	34(CCB), R0		2485
		AB	28	BB40	9E	000CF		MOVAB	@40(CCB)[R0], -76(CCB)		
B4		50	B4	AB	D0	000D5		MOVL	-76(CCB), R0		2493
		26	FF	A0	91	000D9		CMPL	-1(R0), #38		
				05	12	000DD		BNEQ	11\$		
			B4	AB	D7	000DF		DECL	-76(CCB)		2496
				10	11	000E2		BRB	13\$		2497
	97	AB		02	8A	000E4	11\$:	BICB2	#2, -105(CCB)		2500
				0E	11	000E8		BRB	14\$		2505
	B0	AB	28	AB	D0	000EA	12\$:	MOVL	40(CCB), -80(CCB)		2513
	B4	AB	B0	AB	D0	000EF		MOVL	-80(CCB), -76(CCB)		2514
	97	AB		02	88	000F4	13\$:	BISB2	#2, -105(CCB)		2515
				0C	BA	000F8	14\$:	POPR	#^M<R2,R3>		2519
				05	000FA			RSB			

; Routine Size: 251 bytes, Routine Base: \_BASS\$CODE + 0231

; 1235 2520 1



```
1237 2521 1 GLOBAL ROUTINE BAS$$REC_RSL1 ! Read element transmitter
1238 2522 1 : JSB_REC1 =
1239 2523 1
1240 2524 1 !++
1241 2525 1 FUNCTIONAL DESCRIPTION:
1242 2526 1
1243 2527 1 BAS$$REC_RSL1 reads one record if this is a terminal device.
1244 2528 1 Otherwise an error is signalled.
1245 2529 1 Then return start and end+1 of user
1246 2530 1 part of record to be processed as input.
1247 2531 1
1248 2532 1 FORMAL PARAMETERS:
1249 2533 1
1250 2534 1 NONE
1251 2535 1
1252 2536 1 IMPLICIT INPUTS:
1253 2537 1
1254 2538 1 LUB$W_RBUF_SIZE Size of record buffer allocated in OPEN.
1255 2539 1 LUB$A_RBUF_ADR Address of record buffer from OPEN.
1256 2540 1 LUB$V_TERM_DEV flag indicating a terminal device.
1257 2541 1 RAB$L_RBF Pointer to buffer
1258 2542 1 RAB$W_RSZ buffer size
1259 2543 1
1260 2544 1 IMPLICIT OUTPUTS:
1261 2545 1
1262 2546 1 RECOUNT Own storage for RECOUNT function.
1263 2547 1 LUB$A_BUF_PTR points to first char of user part of
1264 2548 1 record buffer.
1265 2549 1 LUB$A_BUF_END points to end+1 of user part of
1266 2550 1 record buffer.
1267 2551 1
1268 2552 1 ROUTINE VALUE:
1269 2553 1
1270 2554 1 NONE
1271 2555 1
1272 2556 1 SIDE EFFECTS:
1273 2557 1
1274 2558 1 Reads next record from file on this logical unit.
1275 2559 1 SIGNALs Insufficient data or any resultant RMS errors.
1276 2560 1 --
1277 2561 1
1278 2562 2 BEGIN
1279 2563 2
1280 2564 2 EXTERNAL REGISTER
1281 2565 2 CCB : REF BLOCK [, BYTE];
1282 2566 2
1283 2567 2 LOCAL
1284 2568 2 RMS STATUS,
1285 2569 2 T_CCB : REF BLOCK [, BYTE];
1286 2570 2
1287 2571 2 LITERAL
1288 2572 2 K_ESCAPE = %X'1B',
1289 2573 2 K_CR = %X'0D';
1290 2574 2
1291 2575 2 !+
1292 2576 2 ! Check to see if this is a terminal device. If this is
1293 2577 2 ! a terminal then do a GET. GETs for terminals are done each time more
```



```
1294 2578 2 data are needed. If this is not a terminal device then error.
1295 2579 2 Read record into buffer using RMS and check for errors
1296 2580 2
1297 2581 2
1298 2582 2
1299 2583 2 IF (NOT .CCB [LUB$V_ANSI]) AND .CCB [LUB$V_TERM_DEV]
1300 2584 2 THEN
1301 2585 2 BEGIN
1302 2586 2 RMS_STATUS = $GET (RAB = .CCB);
1303 2587 2
1304 2588 2 IF .RMS_STATUS EQL RMSS_CONTROLC
1305 2589 2 THEN
1306 2590 2 BAS$$SIGNAL_CTRLC ();
1307 2591 2
1308 2592 2 IF NOT .RMS_STATUS
1309 2593 2 THEN
1310 2594 2 GET_ERROR (K_STOP);
1311 2595 2
1312 2596 2
1313 2597 2 +
1314 2598 2 Return start-1 and end+1 address of record just read
1315 2599 2 LUB$A_BUF_PTR is set to the beginning-1 of the buffer only for BASIC
1316 2600 2 Input. This is seen as a solution to the problem of the user entering
1317 2601 2 <return> as the response to a prompt (null input record) and an empty
1318 2602 2 or depleted buffer which requires another Get.
1319 2603 2 The algorithm:
1320 2604 2 1) Does LUB$A_BUF_PTR = LUB$A_BUF_END?
1321 2605 2 T: The buffer is depleted - another Get is required.
1322 2606 2 2) Add one to LUB$A_BUF_PTR
1323 2607 2 3) Does LUB$A_BUF_PTR = LUB$A_BUF_END?
1324 2608 2 T: Return the default value.
1325 2609 2 4) Scan for the next field.
1326 2610 2
1327 2611 2
1328 2612 2 CCB [LUB$A_BUF_PTR] = .CCB [RAB$L_RBF] - 1;
1329 2613 2 CCB [LUB$A_BUF_END] = .CCB [RAB$L_RBF] + .CCB [RAB$W_RSZ];
1330 2614 2 END
1331 2615 2 ELSE
1332 2616 2
1333 2617 2 +
1334 2618 2 This is not a terminal device
1335 2619 2 Signal insufficient data unless this is an ANSI INPUT.
1336 2620 2 ANSI INPUT errors should cause the statement to be restarted.
1337 2621 2 (This happens in BAS$$HANDLER).
1338 2622 2
1339 2623 2
1340 2624 2 IF NOT .CCB [LUB$V_ANSI]
1341 2625 2 THEN
1342 2626 2 BAS$$SIGNAL (BAS$K_NOTENODAT)
1343 2627 2 ELSE
1344 2628 2 BAS$$SIGNAL_IO (BAS$K_TOOLITDAT);
1345 2629 2
1346 2630 2
1347 2631 2 +
1348 2632 2 Update the cursor position if this input was terminated by an escape.
1349 2633 2 Save cursor position if last PRINT terminator was a semi or comma.
1350 2634 2 Use BUDDY_PTR 'cuz we want to use the PRINT data base for channel 0
```



```
! End of BAS$$REC_RSL1
```

Address	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418	Op419
---------	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------



15	C8 FE	A1 AB 1B		50	D0	00081	7\$:	MOVL	R0, -56(T CCB)		
				04	E1	00085		BBC	#4, -2(CCB), 9\$		2646
				52	B1	0008A		CMPW	R2, #27		2648
				06	12	0008D		BNEQ	8\$		
		50	OE	AB	3C	0008F		MOVZWL	14(CCB), R0		
				0C	11	00093		BRB	10\$		
		0D		52	B1	00095	8\$:	CMPW	R2, #13		2649
				05	12	00098		BNEQ	9\$		
		50		02	D0	0009A		MOVL	#2, R0		
				02	11	0009D		BRB	10\$		
				50	D4	0009F	9\$:	CLRL	R0		2646
		51	22	AB	3C	000A1	10\$:	MOVZWL	34(CCB), R1		
00000000*	EF	50		51	11	000A5		ADDL3	R1, R0, RECOUNT		
		50		01	D0	000AD		MOVL	#1, R0		2652
				04	BA	000B0		POPR	#*M<R2>		2653
				05	000B2			RSB			

; Routine Size: 179 bytes, Routine Base: \_BASS\$CODE + 032C

; 1370 2654 1



```
: 1372      2655 1 GLOBAL ROUTINE BASS$REC_MIN1      ! MAT Input element transmitter
: 1373      2656 1 : JSB_REC1 =
: 1374      2657 1
: 1375      2658 1 ++
: 1376      2659 1 FUNCTIONAL DESCRIPTION:
: 1377      2660 1
: 1378      2661 1     BASS$REC_MIN1 reads one record and checks for a continuation character.
: 1379      2662 1     Then return start and end+1 of user
: 1380      2663 1     part of record to be processed as input.
: 1381      2664 1
: 1382      2665 1 FORMAL PARAMETERS:
: 1383      2666 1
: 1384      2667 1     NONE
: 1385      2668 1
: 1386      2669 1 IMPLICIT INPUTS:
: 1387      2670 1
: 1388      2671 1     LUB$W_RBUF_SIZE      Size of record buffer allocated in OPEN.
: 1389      2672 1     LUB$A_RBUF_ADR      Address of record buffer from OPEN.
: 1390      2673 1     LUB$V_TERM_DEV      flag indicating a terminal device.
: 1391      2674 1     RAB$L_RBF          Pointer to buffer
: 1392      2675 1     RAB$W_RSZ          buffer size
: 1393      2676 1
: 1394      2677 1 IMPLICIT OUTPUTS:
: 1395      2678 1
: 1396      2679 1     RECOUNT          Own storage for RECOUNT function.
: 1397      2680 1     LUB$A_BUF_PTR      points to first char of user part of
: 1398      2681 1     LUB$A_BUF_END      record buffer.
: 1399      2682 1     LUB$A_BUF_END      points to end+1 of user part of
: 1400      2683 1     LUB$A_BUF_END      record buffer.
: 1401      2684 1
: 1402      2685 1 ROUTINE VALUE:
: 1403      2686 1
: 1404      2687 1     NONE
: 1405      2688 1
: 1406      2689 1 SIDE EFFECTS:
: 1407      2690 1
: 1408      2691 1     Reads next record from file on this logical unit.
: 1409      2692 1     SIGNALs any resultant RMS errors.
: 1410      2693 1 --
: 1411      2694 1
: 1412      2695 2 BEGIN
: 1413      2696 2
: 1414      2697 2 EXTERNAL REGISTER
: 1415      2698 2     CCB : REF BLOCK [, BYTE];
: 1416      2699 2
: 1417      2700 2 LITERAL
: 1418      2701 2     K_ESCAPE = %X'1B',
: 1419      2702 2     K_CR = %X'0D';
: 1420      2703 2
: 1421      2704 2 LOCAL
: 1422      2705 2     RMS_STATUS,
: 1423      2706 2     T_CCB : REF BLOCK [, BYTE],
: 1424      2707 2     STATUS;
: 1425      2708 2
: 1426      2709 2
: 1427      2710 2
: 1428      2711 2 !+
: 1428      2711 2 ! Return status to UDF of whether
: 1428      2711 2 ! to keep reading
```

```

1429      2712      2      ! Read record into buffer using RMS and check for errors and a continuation character
1430      2713      2      ! Signal any RMS errors directly.
1431      2714      2      !-
1432      2715      2
1433      2716      2      IF .CCB [ISBSV_MAT_CONT]
1434      2717      2      THEN
1435      2718      2          BEGIN
1436      2719      2              RMS_STATUS = $GET (RAB = .CCB);
1437      2720      2
1438      2721      2              IF .RMS_STATUS EQL RMS$_CONTROL_C
1439      2722      2              THEN
1440      2723      2                  BAS$$SIGNAL_CTRL_C ();
1441      2724      2
1442      2725      2              IF NOT .RMS_STATUS
1443      2726      2              THEN
1444      2727      2                  GET_ERROR (K_STOP);
1445      2728      2
1446      2729      2
1447      2730      2      !+
1448      2731      2      ! Return start-1 and end+1 address of record just read
1449      2732      2      ! LUB$A_BUF_PTR is set to the beginning-1 of the buffer only for BASIC
1450      2733      2      ! Input. This is seen as a solution to the problem of the user entering
1451      2734      2      ! <return> as the response to a prompt (null input record) and an empty
1452      2735      2      ! or depleted buffer which requires another Get.
1453      2736      2      ! The algorithm:
1454      2737      2      ! 1) Does LUB$A_BUF_PTR = LUB$A_BUF_END?
1455      2738      2      !     T: The buffer is depleted - another Get is required.
1456      2739      2      ! 2) Add one to LUB$A_BUF_PTR
1457      2740      2      ! 3) Does LUB$A_BUF_PTR = LUB$A_BUF_END?
1458      2741      2      !     T: Return the default value.
1459      2742      2      ! 4) Scan for the next field.
1460      2743      2      !-
1461      2744      2
1462      2745      2      CCB [LUB$A_BUF_PTR] = .CCB [RAB$_RBF] - 1;
1463      2746      2      CCB [LUB$A_BUF_END] = .CCB [RAB$_RBF] + .CCB [RAB$_RSZ];
1464      2747      2
1465      2748      2      !+
1466      2749      2      ! Check for an '8' as the last character of the record. If it is there,
1467      2750      2      ! it is a continuation character and signifies that there is more data to
1468      2751      2      ! come in the next record.
1469      2752      2      !-
1470      2753      2
1471      2754      2      IF !(.CCB [LUB$A_BUF_END] - 1) < 0, 8> EQLU K_MAT_CONT_CHAR
1472      2755      2      THEN
1473      2756      2          BEGIN
1474      2757      2              CCB [LUB$A_BUF_END] = .CCB [LUB$A_BUF_END] - 1;
1475      2758      2              CCB [ISBSV_MAT_CONT] = 1;
1476      2759      2              END
1477      2760      2          ELSE
1478      2761      2              CCB [ISBSV_MAT_CONT] = 0;
1479      2762      2
1480      2763      2
1481      2764      2      !+
1482      2765      2      ! Update the cursor position if this input was terminated by an escape.
1483      2766      2      ! Save the cursor position if last PRINT terminator was a semi or comma.
1484      2767      2      ! Use BUDDY_PTR 'cuz we want to use the PRINT data base for channel 0
1485      2768      2      !-

```



```
: 1486      2769 3      T_CCB = .CCB [LUB$A BUDDY_PTR];
: 1487      2770 4      T_CCB [LUB$L_PRINT_POS] = -(IF .CCB [RAB$W_STV0] EQL K_ESCAPE AND .T_CCB [LUB$V_FORM_CHAR] EQLU 1
: 1488      2771 4      THEN .CCB [RAB$W_RSZ] + .T_CCB [LUB$L_PRINT_POS] + 1
: 1489      2772 3      ELSE 0);
: 1490      2773 3
: 1491      2774 3
: 1492      2775 3      !+ Set RECOUNT to the number of bytes read
: 1493      2776 3      ! If the file is a terminal format file, then RECOUNT has to be
: 1494      2777 3      ! adjusted for the carriage control terminator.
: 1495      2778 3      !-
: 1496      2779 3
: 1497      2780 4      RECOUNT = .CCB [RAB$W_RSZ] + (IF .CCB [LUB$V_TERM_FOR] THEN SELECTONEU .CCB [RAB$W_STV0] OF
: 1498      2781 4      SET
: 1499      2782 4      [K_ESCAPE] : .CCB [RAB$W_STV2];
: 1500      2783 4      [K_CR] : 2;
: 1501      2784 4      [OTHERWISE] : 0;
: 1502      2785 3      TES ELSE 0);
: 1503      2786 3      STATUS = 1;
: 1504      2787 3      END
: 1505      2788 2      ELSE
: 1506      2789 2      STATUS = 0;
: 1507      2790 2
: 1508      2791 2      RETURN .STATUS;
: 1509      2792 1      END;
```

! End of BASS\$REC\_MIN1

			52	DD	00000	BASS\$REC_MIN1::			
			01	E0	00002	PUSHL	R2		2655
03	97	AB	009E	31	00007	BBS	#1, -105(CCB), 1\$		2716
			5B	DD	0000A	BRW	11\$		
	00000000G	00	01	FB	0000C	PUSHL	CCB		2720
		52	50	D0	00013	CALLS	#1, SYSSGET		
	00010651	8F	52	D1	00016	MOVL	R0, RMS_STATUS		
			07	12	0001D	CMP	RMS_STATUS, #67153		2722
	00000000G	00	00	FB	0001F	BNEQ	2\$		
		07	52	E8	00026	CALLS	#0, BASS\$SIGNAL_CTRL		2724
			7E	D4	00029	BLBS	RMS_STATUS, 3\$		2726
	0000V	CF	01	FB	0002B	CLRL	-(SP)		2728
B0	AB	28	01	C3	00030	CALLS	#1, GET_ERROR		
		50	01	C3	00030	SUBL3	#1, 40(CCB), -80(CCB)		2746
	B4	AB	22	AB	3C	MOVZWL	34(CCB), R0		2747
		50	B4	AB	9E	MOVAB	@40(CCB)[R0], -76(CCB)		
		26	FF	A0	91	MOVL	-76(CCB), R0		2755
			09	12	00048	CMPB	-1(R0), #38		
			B4	AB	D7	BNEQ	4\$		
	97	AB	02	88	0004D	DECL	-76(CCB)		2758
			04	11	00051	BISB2	#2, -105(CCB)		2759
	97	AB	02	8A	00053	BRB	5\$		2755
		50	B8	AB	D0	BICB2	#2, -105(CCB)		2762
		52	OC	AB	3C	MOVL	-72(CCB), T_CCB		2769
		1B	52	B1	0005F	MOVZWL	12(CCB), R2		2770
			11	12	00062	CMPW	R2, #27		
OC	FE	A0	02	E1	00064	BNEQ	6\$		
						BBC	#2, -2(T_CCB), 6\$		

		51	22	AB	3C	00069	MOVZWL	34(CCB), R1	2771
		51	C8	A0	C0	0006D	ADDL2	-56(T_CCB), R1	
				51	D6	00071	INCL	R1	
				02	11	00073	BRB	7\$	
				51	D4	00075	CLRL	R1	2770
15	C8	A0		51	D0	00077	MOVL	R1, -56(T_CCB)	
	FE	AB		04	E1	0007B	BBC	#4, -2(CCB), 9\$	2780
		1B		52	B1	00080	CMPW	R2, #27	2782
				06	12	00083	BNEQ	8\$	
		50	0E	AB	3C	00085	MOVZWL	14(CCB), R0	
				0C	11	00089	BRB	10\$	
		0D		52	B1	0008B	CMPW	R2, #13	2783
				05	12	0008E	BNEQ	9\$	
		50		02	D0	00090	MOVL	#2, R0	
				02	11	00093	BRB	10\$	
				50	D4	00095	CLRL	R0	2780
00000000' EF		51	22	AB	3C	00097	MOVZWL	34(CCB), R1	
		50		51	C1	0009B	ADDL3	R1, R0, RECOUNT	
		50		01	D0	000A3	MOVL	#1, STATUS	2786
				02	11	000A6	BRB	12\$	2716
				50	D4	000A8	CLRL	STATUS	2789
				04	BA	000AA	POPR	#^M<R2>	2792
				05	000AC		RSB		

; Routine Size: 173 bytes, Routine Base: \_BAS\$CODE + 03DF

; 1510 2793 1



```

: 1512      2794 1 GLOBAL ROUTINE BAS$$REC_RSL9          ! Read IO_END
: 1513      2795 1   : JSB_REC9 NOVALUE =
: 1514      2796 1
: 1515      2797 1   ++
: 1516      2798 1   FUNCTIONAL DESCRIPTION:
: 1517      2799 1
: 1518      2800 1       BAS$$REC_RSL9 is a no-op!
: 1519      2801 1
: 1520      2802 1   FORMAL PARAMETERS:
: 1521      2803 1
: 1522      2804 1       NONE
: 1523      2805 1
: 1524      2806 1   IMPLICIT INPUTS:
: 1525      2807 1
: 1526      2808 1       NONE
: 1527      2809 1
: 1528      2810 1   IMPLICIT OUTPUTS:
: 1529      2811 1
: 1530      2812 1   ROUTINE VALUE:
: 1531      2813 1
: 1532      2814 1       NONE
: 1533      2815 1
: 1534      2816 1   SIDE EFFECTS:
: 1535      2817 1
: 1536      2818 1   --
: 1537      2819 1
: 1538      2820 2   BEGIN
: 1539      2821 2   RETURN;
: 1540      2822 1   END;                                ! End of BAS$$_REC_RSL9

```

05 00000 BAS\$\$REC\_RSL9::  
RSB

; 2822

; Routine Size: 1 bytes, Routine Base: \_BAS\$CODE + 048C

; 1541 2823 1

```

: 1543      2824 1 GLOBAL ROUTINE BAS$$REC_MIN9      ! MAT Input IO_END
: 1544      2825 1   : JSB_REC9 NOVALUE =
: 1545      2826 1
: 1546      2827 1 ++
: 1547      2828 1 FUNCTIONAL DESCRIPTION:
: 1548      2829 1
: 1549      2830 1     BAS$$REC_RSL9 is a no-op!
: 1550      2831 1
: 1551      2832 1 FORMAL PARAMETERS:
: 1552      2833 1
: 1553      2834 1     NONE
: 1554      2835 1
: 1555      2836 1 IMPLICIT INPUTS:
: 1556      2837 1
: 1557      2838 1     NONE
: 1558      2839 1
: 1559      2840 1 IMPLICIT OUTPUTS:
: 1560      2841 1
: 1561      2842 1 ROUTINE VALUE:
: 1562      2843 1
: 1563      2844 1     NONE
: 1564      2845 1
: 1565      2846 1 SIDE EFFECTS:
: 1566      2847 1
: 1567      2848 1 --
: 1568      2849 1
: 1569      2850 2 BEGIN
: 1570      2851 2 RETURN;
: 1571      2852 1 END;                                ! End of BAS$$REC_MIN9

```

05 00000 BAS\$\$REC\_MIN9::  
RSB

; 2852

; Routine Size: 1 bytes, Routine Base: \_BAS\$CODE + 048D

; 1572 2853 1



```
: 1574      2854 1 GLOBAL ROUTINE BASS$REC_MLI1      ! MAT Linput element transmitter
: 1575      2855 1 : JSB_REC1 =
: 1576      2856 1
: 1577      2857 1 !++
: 1578      2858 1 FUNCTIONAL DESCRIPTION:
: 1579      2859 1
: 1580      2860 1     BASS$REC_MLI1 unconditionally reads one record. There is no
: 1581      2861 1     continuation character for MAT LINPUT.
: 1582      2862 1     Otherwise an error is signalled.
: 1583      2863 1     Then return start and end+1 of user
: 1584      2864 1     part of record to be processed as input.
: 1585      2865 1
: 1586      2866 1 FORMAL PARAMETERS:
: 1587      2867 1
: 1588      2868 1     NONE
: 1589      2869 1
: 1590      2870 1 IMPLICIT INPUTS:
: 1591      2871 1
: 1592      2872 1     LUB$W_RBUF_SIZE      Size of record buffer allocated in OPEN.
: 1593      2873 1     LUB$A_RBUF_ADR      Address of record buffer from OPEN.
: 1594      2874 1     RAB$L_RBF        Pointer to buffer
: 1595      2875 1     RAB$W_RSZ        buffer size
: 1596      2876 1
: 1597      2877 1 IMPLICIT OUTPUTS:
: 1598      2878 1
: 1599      2879 1     RECOUNT      Own storage for RECOUNT function.
: 1600      2880 1     LUB$A_BUF_PTR    points to first char of user part of
: 1601      2881 1
: 1602      2882 1     LUB$A_BUF_END    points to end+1 of user part of
: 1603      2883 1
: 1604      2884 1
: 1605      2885 1 ROUTINE VALUE:
: 1606      2886 1
: 1607      2887 1     NONE
: 1608      2888 1
: 1609      2889 1 SIDE EFFECTS:
: 1610      2890 1
: 1611      2891 1     Reads next record from file on this logical unit.
: 1612      2892 1     SIGNALS any resultant RMS errors.
: 1613      2893 1 --
: 1614      2894 1
: 1615      2895 2 BEGIN
: 1616      2896 2
: 1617      2897 2 EXTERNAL REGISTER
: 1618      2898 2     CCB : REF BLOCK [, BYTE];
: 1619      2899 2
: 1620      2900 2 LOCAL
: 1621      2901 2     RMS STATUS,
: 1622      2902 2     T_CCB : REF BLOCK [, BYTE];
: 1623      2903 2
: 1624      2904 2 LITERAL
: 1625      2905 2     K_ESCAPE = %X'1B',
: 1626      2906 2     K_CR = %X'0D';
: 1627      2907 2
: 1628      2908 2 !+
: 1629      2909 2 ! Read record into buffer using RMS and check for errors
: 1630      2910 2 ! Signal any RMS errors directly.
```

```
1631 2911 2 :-
1632 2912 2
1633 2913 2 RMS_STATUS = $GET (RAB = .CCB);
1634 2914 2
1635 2915 2 IF .RMS_STATUS EQL RMSS$CONTROL.C
1636 2916 2 THEN
1637 2917 2     BAS$$SIGNAL_CTRL_C ();
1638 2918 2
1639 2919 2 IF NOT .RMS_STATUS
1640 2920 2 THEN
1641 2921 2     GET_ERROR (K_STOP);
1642 2922 2
1643 2923 2 +
1644 2924 2 Return start-1 and end+1 address of record just read
1645 2925 2 LUB$A_BUF_PTR is set to the beginning-1 of the buffer only for BASIC
1646 2926 2 Input. This is seen as a solution to the problem of the user entering
1647 2927 2 <return> as the response to a prompt (null input record) and an empty
1648 2928 2 or depleted buffer which requires another Get.
1649 2929 2 The algorithm:
1650 2930 2 1) Does LUB$A_BUF_PTR = LUB$A_BUF_END?
1651 2931 2    T: The buffer is depleted - another Get is required.
1652 2932 2 2) Add one to LUB$A_BUF_PTR
1653 2933 2 3) Does LUB$A_BUF_PTR = LUB$A_BUF_END?
1654 2934 2    T: Return the default value.
1655 2935 2 4) Scan for the next field.
1656 2936 2
1657 2937 2 -
1658 2938 2 CCB [LUB$A_BUF_PTR] = .CCB [RAB$L_RBF] - 1;
1659 2939 2 CCB [LUB$A_BUF_END] = .CCB [RAB$L_RBF] + .CCB [RAB$W_RSZ];
1660 2940 2 +
1661 2941 2 Update the cursor position if this input was terminated by an escape.
1662 2942 2 Save the cursor position if last PRINT terminator was a semi or comma.
1663 2943 2 Use BUDDY_PTR 'cuz we want to use the PRINT data base for channel 0
1664 2944 2 -
1665 2945 2 T_CCB = .CCB [LUB$A_BUDDY_PTR];
1666 2946 2 T_CCB [LUB$L_PRINT_POS] = (IF .CCB [RAB$W_STV0] EQL K_ESCAPE AND .T_CCB [LUB$V_FORM_CHAR] EQLU 1
1667 2947 2     THEN .CCB [RAB$W_RSZ] + .T_CCB [LUB$L_PRINT_POS] + 1
1668 2948 2     ELSE 0);
1669 2949 2 +
1670 2950 2 Set RECOUNT to the number of bytes read
1671 2951 2 If the file is a terminal format file, then RECOUNT has to be
1672 2952 2 adjusted for the carriage control terminator.
1673 2953 2 -
1674 2954 2 RECOUNT = .CCB [RAB$W_RSZ] + (IF .CCB [LUB$V_TERM_FOR] THEN SELECTONEU .CCB [RAB$W_STV0] OF
1675 2955 2     SET
1676 2956 2     [K_ESCAPE] : .CCB [RAB$W_STV2];
1677 2957 2     [K_CR] : 2;
1678 2958 2     [OTHERWISE] : 0;
1679 2959 2     TES ELSE 0);
1680 2960 2 RETURN 1
1681 2961 1 END;

! End of BAS$$REC_ML11
```



					5B	DD	00002	PUSHL	R2		2854
		00000000G	00		01	FB	00004	PUSHL	CCB		2913
			52		50	D0	0000B	CALLS	#1, SYSSGET		
		00010651	8F		52	D1	0000E	MOVL	R0, RMS_STATUS		
					07	12	00015	CMPL	RMS_STATUS, #67153		2915
		00000000G	00		00	FB	00017	BNEQ	1\$		
			07		52	E8	0001E	CALLS	#0, BASS\$SIGNAL_CTRL		2917
					7E	D4	00021	BLBS	RMS_STATUS, 2\$		2919
		0000V	CF		01	FB	00023	CLRL	-(SP)		2921
B0	AB	28	AB		01	C3	00028	CALLS	#1, GET_ERROR		
			50		22	AB	3C	SUBL3	#1, 40(CCB), -80(CCB)		2938
		B4	AB		28	BB	40	MOVZWL	34(CCB), R0		2939
			51		B8	AB	D0	MOVAB	@40(CCB)[R0], -76(CCB)		
			52		OC	AB	3C	MOVL	-72(CCB), T_CCB		2945
			1B			52	B1	MOVZWL	12(CCB), R2		2946
						11	12	CMPL	R2, #27		
						02	E1	BNEQ	3\$		
	OC	FE	A1			AB	3C	BBC	#2, -2(T_CCB), 3\$		
			50		22	A1	C0	MOVZWL	34(CCB), R0		2947
			50		C8	50	D6	ADDL2	-56(T_CCB), R0		
						02	11	INCL	R0		
						50	D4	BRB	4\$		
						50	D0	CLRL	R0		2946
		C8	A1			04	E1	MOVL	R0, -56(T_CCB)		
	15	FE	AB			52	B1	BBC	#4, -2(CCB), 6\$		2954
			1B			06	12	CMPL	R2, #27		2956
			50		OE	AB	3C	BNEQ	5\$		
						OC	11	MOVZWL	14(CCB), R0		
			OD			52	B1	BRB	7\$		
			50			05	12	CMPL	R2, #13		2957
						02	D0	BNEQ	6\$		
						02	11	MOVL	#2, R0		
						50	D4	BRB	7\$		
			51		22	AB	3C	CLRL	R0		2954
00000000*	EF		50			51	C1	MOVZWL	34(CCB), R1		
			50			01	D0	ADDL3	R1, R0, RECOUNT		2960
						04	BA	MOVL	#1, R0		2961
						05	00089	POPR	#*M<R2>		
								RSB			

; Routine Size: 138 bytes, Routine Base: \_BAS\$CODE + 048E

; 1682 2962 1

```
1684 2963 1 GLOBAL ROUTINE BASS$REC_WSLO ! Write list-directed
1685 2964 1 : JSB_RECO NOVALUE =
1686 2965 1
1687 2966 1 !++
1688 2967 1 FUNCTIONAL DESCRIPTION:
1689 2968 1
1690 2969 1 BASS$REC_WSLO prepares a record for list-directed output.
1691 2970 1 Then return start and end+1 of user
1692 2971 1 part of record to be processed.
1693 2972 1
1694 2973 1 FORMAL PARAMETERS:
1695 2974 1
1696 2975 1 NONE
1697 2976 1
1698 2977 1 IMPLICIT INPUTS:
1699 2978 1
1700 2979 1 LUB$W_RBUF_SIZE Size (bytes) allocated for record buffer at OPEN.
1701 2980 1 LUB$A_RBUF_ADR Address of record buffer allocated at OPEN
1702 2981 1 LUB$V_FIXED 1 if fixed-length records
1703 2982 1 LUB$V_FORM_CHAR Indicates that the last element transmitter ended
1704 2983 1 in a comma or semicolon format char.
1705 2984 1 LUB$V_FORCIBLE Indicates a forcible device
1706 2985 1 LUB$V_CCO Cancel control 0
1707 2986 1
1708 2987 1 IMPLICIT OUTPUTS:
1709 2988 1
1710 2989 1 LUB$B_BAS_VFC1 'Pre' carriage control
1711 2990 1 LUB$B_BAS_VFC2 'Post' carriage control
1712 2991 1 LUB$A_BUF_PTR pointer to next byte of buffer
1713 2992 1 LUB$A_BUF_END pointer to byte following the buffer
1714 2993 1 RAB$V_CCO Cancel control 0
1715 2994 1
1716 2995 1 ROUTINE VALUE:
1717 2996 1
1718 2997 1 NONE
1719 2998 1
1720 2999 1 SIDE EFFECTS:
1721 3000 1
1722 3001 1 --
1723 3002 1
1724 3003 2 BEGIN
1725 3004 2
1726 3005 2 EXTERNAL REGISTER
1727 3006 2 CCB : REF BLOCK [, BYTE];
1728 3007 2
1729 3008 2 !+
1730 3009 2 Copy the current status of the cancel-control-o bit in the LUB
1731 3010 2 (possibly set by RCTRL0) into the RAB, and clear it from the
1732 3011 2 LUB. The net effect of this is that if the bit is set in the
1733 3012 2 LUB, then the CANCTRL0 modifier will be applied to this write
1734 3013 2 operation only.
1735 3014 2
1736 3015 2
1737 3016 2 CCB [RAB$V_CCO] = .CCB [LUB$V_CCO];
1738 3017 2 CCB [LUB$V_CCO] = 0;
1739 3018 2
1740 3019 2 !+
```



```
: 1741      3020  2  ! If there is a format character pending and this is not a forcible
: 1742      3021  2  ! device, then don't change the buffer pointers. The PUT will be done when
: 1743      3022  2  ! there is no format character pending.
: 1744      3023  2  !-
: 1745      3024  2  !-
: 1746      3025  2  IF .CCB [LUB$V_FORM_CHAR] AND NOT .CCB [LUB$V_FORCIBLE] THEN RETURN;
: 1747      3026  2  !-
: 1748      3027  2  !+
: 1749      3028  2  ! If the last statement did not end with a format character,
: 1750      3029  2  ! then put a line feed into the 'pre' carriage control
: 1751      3030  2  ! Unconditionally set the 'post' carriage control to null
: 1752      3031  2  !-
: 1753      3032  2  !-
: 1754      3033  2  CCB [LUB$B_BAS_VFC1] = (IF .CCB [LUB$V_FORM_CHAR] THEN BASSK_NULL ELSE BASSK_LF);
: 1755      3034  2  CCB [LUB$B_BAS_VFC2] = BASSK_NULL;
: 1756      3035  2  !-
: 1757      3036  2  !+
: 1758      3037  2  ! If the buffer is dirty, then this is recursive I/O and we want to
: 1759      3038  2  ! concatenate this record. So leave the buffer pointers alone. Otherwise
: 1760      3039  2  ! return the buffer pointers initialized for another statement
: 1761      3040  2  !-
: 1762      3041  2  !-
: 1763      3042  2  IF NOT .CCB [LUB$V_OUTBUF_DR]
: 1764      3043  2  THEN
: 1765      3044  3  BEGIN
: 1766      3045  3  CCB [LUB$A_BUF_PTR] = .CCB [LUB$A_RBUF_ADR];
: 1767      3046  3  CCB [LUB$A_BUF_END] = .CCB [LUB$A_RBUF_ADR] + .CCB [LUB$W_RBUF_SIZE];
: 1768      3047  2  END;
: 1769      3048  2  !-
: 1770      3049  2  RETURN;
: 1771      3050  1  END;
```

! END OF BAS\$\$REC\_WSL0

50	A0	AB	01	02	EF 00000	BAS\$\$REC WSL0::			
						EXTZV	#2, #1, -96(CCB), R0		3016
07	AB	01	07	50	F0 00006	INSV	R0, #7, #1, 7(CCB)		
		A0	AB	04	8A 0000C	BICB2	#4, -96(CCB)		3017
		0E	FE	02	E1 00010	BBC	#2, -2(CCB), 1\$		3025
		24	FE	06	E1 00015	BBC	#6, -2(CCB), 3\$		
		04	FE	02	E1 0001A	BBC	#2, -2(CCB), 1\$		3033
				50	D4 0001F	CLRL	R0		
			50	03	11 00021	BRB	2\$		
			01	D0	00023 1\$:	MOVL	#1, R0		
		DA	AB	50	9B 00026 2\$:	MOVZBW	R0, -38(CCB)		
	OF	FE	AB	03	E0 0002A	BBS	#3, -2(CCB), 3\$		3042
		B0	AB	EC	AB D0 0002F	MOVL	-20(CCB), -80(CCB)		3045
			50	D2	AB 3C 00034	MOVZWL	-46(CCB), R0		3046
		B4	AB	EC	BB40 9E 00038	MOVAB	a-20(CCB)[R0], -76(CCB)		
				05	0003E 3\$:	RSB			3050

; Routine Size: 63 bytes, Routine Base: \_BAS\$CODE + 0518

; 1772 3051 1



```
1774 3052 1 GLOBAL ROUTINE BASS$REC_WSL1 ( ! Write list-directed
1775 3053 1 CARRIAGE_CTRL) : JSB_REC_WSL1 NOVALUE = ! Called from BASS$DO_WRITE
1776 3054 1
1777 3055 1 ++
1778 3056 1 FUNCTIONAL DESCRIPTION:
1779 3057 1
1780 3058 1 Write one list-directed record and initialize for the next
1781 3059 1 BASS$REC_WSL1 writes one output buffer and then
1782 3060 1 initializes the output buffer and returns start and end+1 of user
1783 3061 1 part of record buffer to be filled by caller.
1784 3062 1 If this routine is called because the buffer overflowed then the 'post'
1785 3063 1 carriage control should be null. If this routine is called because the
1786 3064 1 margin overflowed, then the 'post' carriage control should be 'CR'.
1787 3065 1
1788 3066 1 FORMAL PARAMETERS:
1789 3067 1
1790 3068 1 CARRIAGE_CTRL.rlu.v carriage control for the record
1791 3069 1
1792 3070 1 IMPLICIT INPUTS:
1793 3071 1
1794 3072 1 LUB$W_RBUF_SIZE Size (bytes) allocated for record buffer at OPEN.
1795 3073 1 LUB$A_RBUF_ADR Address of record buffer allocated at OPEN
1796 3074 1 LUB$A_BUF_PTR Pointer to next byte in user buffer.
1797 3075 1 RAB$L_RBF Pointer to user buffer
1798 3076 1
1799 3077 1 IMPLICIT OUTPUTS:
1800 3078 1
1801 3079 1 LUB$B_BAS_VFC1 'Pre' carriage control
1802 3080 1 LUB$B_BAS_VFC2 'Post' carriage control
1803 3081 1 LUB$A_BUF_PTR Pointer to start of user part of record buffer
1804 3082 1 LUB$A_BUF_END Pointer to end+1 of user part of record buffer
1805 3083 1 LUB$V_OUTBUF_DR Indicates valid data in the output buffer
1806 3084 1 RAB$W_RSZ Size of user buffer
1807 3085 1
1808 3086 1 ROUTINE VALUE:
1809 3087 1
1810 3088 1 NONE
1811 3089 1
1812 3090 1 SIDE EFFECTS:
1813 3091 1
1814 3092 1 Writes one RMS sequential record.
1815 3093 1 SIGNALs BASS$K_FATSYSIO on PUT error.
1816 3094 1 --
1817 3095 1
1818 3096 2 BEGIN
1819 3097 2
1820 3098 2 EXTERNAL REGISTER
1821 3099 2 CCB = 11 : REF BLOCK [, BYTE];
1822 3100 2
1823 3101 2 LITERAL
1824 3102 2 K_NO_CR = 2;
1825 3103 2
1826 3104 2 LOCAL
1827 3105 2 RMS_STATUS;
1828 3106 2
1829 3107 2 ++
1830 3108 2 : Set 'post' carriage control to CR or NULL depending on whether the margin
```



```

1831 3109 2 ! overflowed or the buffer overflowed.
1832 3110 2 ! If this is a file, the carriage control is ignored and a record is PUT.
1833 3111 2 -
1834 3112 2 CCB [LUB$B_BAS_VFC2] = (IF .CARRIAGE_CTRL EQL BAS$K_BUF_EXC THEN BAS$K_NULL ELSE BAS$K_CR);
1835 3113 2
1836 3114 2 !+
1837 3115 2 ! perform the record write.
1838 3116 2 ! Set recordsize to actual length of record
1839 3117 2 !-
1840 3118 2
1841 3119 2 CCB [RAB$W_RSZ] = .CCB [LUB$A_BUF_PTR] - .CCB [LUB$A_BUF_BEG];
1842 3120 2
1843 3121 2 !+
1844 3122 2 ! Output buffer to RMS and check for errors
1845 3123 2 ! If errors, SIGNAL_STO
1846 3124 2 !-
1847 3125 2
1848 3126 2 CCB [RAB$L_RBF] = .CCB [LUB$A_BUF_BEG];
1849 3127 2 CCB [LUB$V_OUTBUF_DR] = 0;
1850 3128 2
1851 3129 2 RMS_STATUS = $PUT (RAB = .CCB);
1852 3130 2
1853 3131 2 IF .RMS_STATUS EQL RM$S_CONTROLC
1854 3132 2 THEN
1855 3133 2 BAS$$SIGNAL_CTRLC ();
1856 3134 2
1857 3135 2 IF NOT .RMS_STATUS
1858 3136 2 THEN
1859 3137 2 PUT_ERROR (K_STOP);
1860 3138 2
1861 3139 2 !+
1862 3140 2 ! Set the 'pre' carriage control to LF if CARRIAGE_CTRL warrants it.
1863 3141 2 ! Set the 'post' carriage control to null.
1864 3142 2 !-
1865 3143 2
1866 3144 2 CCB [LUB$B_BAS_VFC1] = (IF .CARRIAGE_CTRL EQL BAS$K_BUF_EXC THEN BAS$K_NULL ELSE BAS$K_LF);
1867 3145 2 CCB [LUB$B_BAS_VFC2] = BAS$K_NULL;
1868 3146 2
1869 3147 2 !+
1870 3148 2 ! Initialize record buffer for another list-directed write
1871 3149 2 ! return record buffer pointers to caller
1872 3150 2 !-
1873 3151 2
1874 3152 2 CCB [LUB$A_BUF_PTR] = .CCB [LUB$A_RBUF_ADR];
1875 3153 2 CCB [LUB$A_BUF_END] = .CCB [LUB$A_RBUF_ADR] + .CCB [LUB$W_RBUF_SIZE];
1876 3154 2 RETURN;
1877 3155 2 END;

```

! End of routine - BAS\$\$UDF\_WSL1

```

0C BB 00000 BAS$$REC WSL1::
08 53 D4 00002 PUSH R3
50 D1 00004 CLRL R3
06 12 00007 CMPL CARRIAGE_CTRL, #8
BNEQ 1$

```

: 3052  
: 3112  
:

22	AB	DB	50	8D	53	D6	00009	INCL	R3		
		B0	AB		50	D4	0000B	CLRL	R0		
		28	AB	BC	04	11	0000D	BRB	2\$		
		FE	AB		8F	9A	0000F	MOVZBL	#141, R0		
					50	90	00013	MOVB	R0, -37(CCB)		
					AB	A3	00017	SUBW3	-68(CCB), -80(CCB), 34(CCB)		3119
					AB	D0	0001E	MOVL	-68(CCB), 40(CCB)		3126
					08	8A	00023	BICB2	#8, -2(CCB)		3127
					5B	DD	00027	PUSHL	CCB		3129
		00000000G	00		01	FB	00029	CALLS	#1, SYSSPUT		
			52		50	D0	00030	MOVL	R0, RMS_STATUS		
		00010651	8F		52	D1	00033	CMPL	RMS_STATUS, #67153		3131
					07	12	0003A	BNEQ	3\$		
		00000000G	00		00	FB	0003C	CALLS	#0, BASS\$SIGNAL_CTRL C		3133
			07		52	E8	00043	BLBS	RMS_STATUS, 4\$		3135
					7E	D4	00046	CLRL	-(SP)		3137
		0000V	CF		01	FB	00048	CALLS	#1, PUT_ERROR		
			04		53	E9	0004D	BLBC	R3, 5\$		3144
					50	D4	00050	CLRL	R0		
					03	11	00052	BRB	6\$		
					01	D0	00054	MOVL	#1, R0		
		DA	50		50	9B	00057	MOVZBW	R0, -38(CCB)		
		B0	AB	EC	AB	D0	0005B	MOVL	-20(CCB), -80(CCB)		3152
				D2	AB	3C	00060	MOVZWL	-46(CCB), R0		3153
		B4	50	EC	BB	40	9E	MOVAB	@-20(CCB)[R0], -76(CCB)		
			AB		0C	BA	0006A	POPR	#*M<R2,R3>		3155
					05	0006C		RSB			

; Routine Size: 109 bytes, Routine Base: \_BASS\$CODE + 0557

; 1878 3156 1



```
: 1880 3157 1 GLOBAL ROUTINE BASS$REC_RMFO ! Initialize read memory formatted
: 1881 3158 1 : JSB_REC0 NOVALUE =
: 1882 3159 1
: 1883 3160 1 ++
: 1884 3161 1 FUNCTIONAL DESCRIPTION:
: 1885 3162 1
: 1886 3163 1 Pick up pointer to last major frame from ISB and initialize BUF_BEG,
: 1887 3164 1 BUF_PTR, and BUF_END to the values for the data area found in the
: 1888 3165 1 frame.
: 1889 3166 1
: 1890 3167 1 FORMAL PARAMETERS:
: 1891 3168 1
: 1892 3169 1 NONE
: 1893 3170 1
: 1894 3171 1 IMPLICIT INPUTS:
: 1895 3172 1
: 1896 3173 1 ISB$A_MAJ_F_PTR pointer to last Basic major frame
: 1897 3174 1
: 1898 3175 1 IMPLICIT OUTPUTS:
: 1899 3176 1
: 1900 3177 1 ROUTINE VALUE:
: 1901 3178 1
: 1902 3179 1 NONE
: 1903 3180 1
: 1904 3181 1 SIDE EFFECTS:
: 1905 3182 1
: 1906 3183 1 NONE
: 1907 3184 1
: 1908 3185 1 --
: 1909 3186 1
: 1910 3187 2 BEGIN
: 1911 3188 2
: 1912 3189 2 EXTERNAL REGISTER
: 1913 3190 2 CCB = K_CCB_REG : REF BLOCK [, BYTE];
: 1914 3191 2
: 1915 3192 2 LOCAL
: 1916 3193 2 BMF : REF BLOCK [0, BYTE] FIELD (BSF$MAJOR_FRAME);
: 1917 3194 2
: 1918 3195 2 ++
: 1919 3196 2 Reach back into the last major frame by picking up the value of R11 stored
: 1920 3197 2 in the ISB. Initialize BUF_PTR, BUF_END, BUF_BEG so that this will look
: 1921 3198 2 like a vanilla INPUT.
: 1922 3199 2 --
: 1923 3200 2
: 1924 3201 2 BMF = .CCB [ISB$A_MAJ_F_PTR];
: 1925 3202 2 ++
: 1926 3203 2 If this cell is zero, then there was no DATA statement and an error should be
: 1927 3204 2 signalled.
: 1928 3205 2 --
: 1929 3206 2
: 1930 3207 2 IF .BMF [BSF$A_CUR_DTA] EQLA 0 THEN BASS$STOP_IO (BASS$K_OUTOF_DAT);
: 1931 3208 2
: 1932 3209 2 CCB [LUB$A_BUF_BEG] = .BMF [BSF$A_CUR_DTA];
: 1933 3210 2 CCB [LUB$A_BUF_END] = .BMF [BSF$A_END_DTA];
: 1934 3211 2
: 1935 3212 2 ++
: 1936 3213 2 Subtract one from CUR_DATA for INPUT element transmitter compatibility.
```

BAS\$\$REC\_PROC  
1-095

F 8  
16-Sep-1984 01:01:12  
14-Sep-1984 11:56:35

VAX-11 Bliss-32 V4.0-742  
[BASRTL.SRC]BASRECPRO.B32;1

Page 52  
(20)

```
: 1937      3214  2    !-
: 1938      3215  2
: 1939      3216  2    CCB [LUB$A_BUF_PTR] = .BMF [BSF$A_CUR_DTA] - 1;
: 1940      3217  2    RETURN;
: 1941      3218  1    END;
```

```
                                OC  BB 00000 BAS$$REC RMF0::
                                52    FF48 CB D0 00002    PUSHR    #^M<R2,R3>
                                53    0087 C2 D0 00007    MOVL     -184(CCB), BMF
                                7E    00G  0B 12 0000C    MOVL     135(BMF), R3
                                00000000G 00 8F 9A 0000E    BNEQ     1$
                                BC  AB    01 FB 00012    MOVZBL  #BAS$K_OUTOF_DAT, -(SP)
                                B4  AB    53 D0 00019 1$:    CALLS   #1, BAS$$STOP_IO
                                B0  AB    C2 D0 0001D    MOVL     R3, -68(CCB)
                                FF    A3 9E 00023    MOVL     139(BMF), -76(CCB)
                                OC    BA 00028    MOVAB   -1(R3), -80(CCB)
                                05 0002A    POPR    #^M<R2,R3>
                                RSB
                                : 3157
                                : 3201
                                : 3207
                                :
                                :
                                : 3209
                                : 3210
                                : 3216
                                : 3218
                                :
```

; Routine Size: 43 bytes, Routine Base: \_BAS\$CODE + 05C4

: 1942 3219 1



```

: 1944      3220 1 GLOBAL ROUTINE BAS$$REC_MRE1      ! Mat Read element transmitter
: 1945      3221 1   : JSB_REC1 =
: 1946      3222 1
: 1947      3223 1 ++
: 1948      3224 1 FUNCTIONAL DESCRIPTION:
: 1949      3225 1
: 1950      3226 1     Since MAT READ just takes as much input data as it can get, it will just
: 1951      3227 1     return a failure here because there is no more data.
: 1952      3228 1
: 1953      3229 1 FORMAL PARAMETERS:
: 1954      3230 1
: 1955      3231 1     NONE
: 1956      3232 1
: 1957      3233 1 IMPLICIT INPUTS:
: 1958      3234 1
: 1959      3235 1     NONE
: 1960      3236 1
: 1961      3237 1 IMPLICIT OUTPUTS:
: 1962      3238 1
: 1963      3239 1     NONE
: 1964      3240 1
: 1965      3241 1 ROUTINE VALUE:
: 1966      3242 1
: 1967      3243 1     Returns failure - out of data.
: 1968      3244 1
: 1969      3245 1 SIDE EFFECTS:
: 1970      3246 1
: 1971      3247 1     As a result of the failure being returned, the MAT READ will stop
: 1972      3248 1     filling the matrix.
: 1973      3249 1
: 1974      3250 1 --
: 1975      3251 1
: 1976      3252 2 BEGIN
: 1977      3253 2 RETURN 0
: 1978      3254 1 END;

```

! end of BAS\$\$REC\_MRE1

```

50 D4 00000 BAS$$REC_MRE1::
    CLRL R0
05 00002 RSB

```

: 3253  
: 3254

; Routine Size: 3 bytes, Routine Base: \_BAS\$CODE + 05EF

; 1979 3255 1

```
: 1981      3256 1 GLOBAL ROUTINE BASS$REC_RMF1          ! Read element transmitter
: 1982      3257 1   : JSB_REC1 NOVALUE =
: 1983      3258 1
: 1984      3259 1   ++
: 1985      3260 1 FUNCTIONAL DESCRIPTION:
: 1986      3261 1
: 1987      3262 1       BASS$REC_RMF1 should not be called in normal processing and will signal
: 1988      3263 1       an error (BASS$K_OUTOF_DAT) if it is called.
: 1989      3264 1
: 1990      3265 1 FORMAL PARAMETERS:
: 1991      3266 1
: 1992      3267 1     NONE
: 1993      3268 1
: 1994      3269 1 IMPLICIT INPUTS:
: 1995      3270 1
: 1996      3271 1     NONE
: 1997      3272 1
: 1998      3273 1 IMPLICIT OUTPUTS:
: 1999      3274 1
: 2000      3275 1 ROUTINE VALUE:
: 2001      3276 1
: 2002      3277 1     NONE
: 2003      3278 1
: 2004      3279 1 SIDE EFFECTS:
: 2005      3280 1
: 2006      3281 1     Signal - BASS$K_OUTOF_DAT - Out of Data
: 2007      3282 1
: 2008      3283 1   --
: 2009      3284 1
: 2010      3285 2 BEGIN
: 2011      3286 2 BASS$$SIGNAL (BASS$K_OUTOF_DAT);
: 2012      3287 2 RETURN;
: 2013      3288 1 END;                                ! end of BASS$REC_RMF1
```

```
              7E      00G  8F  9A 00000 BASS$REC_RMF1::
              00000000G 00      01  FB 00004  MOVZBL #BASS$K_OUTOF_DAT, -(SP)
                                05 0000B  CALLS  #1, BASS$$SIGNAL
                                RSB
```

```
: 3286
: 3288
```

; Routine Size: 12 bytes, Routine Base: \_BASS\$CODE + 05F2

; 2014 3289 1



```
: 2016      3290 1 GLOBAL ROUTINE BAS$$REC_RMF9      ! Read IO_END
: 2017      3291 1 : JSB_REC9 NOVALUE =
: 2018      3292 1
: 2019      3293 1 ++
: 2020      3294 1 FUNCTIONAL DESCRIPTION:
: 2021      3295 1
: 2022      3296 1      Update the current data pointer in the last Basic major frame
: 2023      3297 1
: 2024      3298 1 FORMAL PARAMETERS:
: 2025      3299 1
: 2026      3300 1      NONE
: 2027      3301 1
: 2028      3302 1 IMPLICIT INPUTS:
: 2029      3303 1
: 2030      3304 1      NONE
: 2031      3305 1
: 2032      3306 1 IMPLICIT OUTPUTS:
: 2033      3307 1
: 2034      3308 1 ROUTINE VALUE:
: 2035      3309 1
: 2036      3310 1      NONE
: 2037      3311 1
: 2038      3312 1 SIDE EFFECTS:
: 2039      3313 1
: 2040      3314 1 --
: 2041      3315 1
: 2042      3316 2 BEGIN
: 2043      3317 2
: 2044      3318 2 EXTERNAL REGISTER
: 2045      3319 2      CCB = K_CCB_REG : REF BLOCK [0, BYTE];
: 2046      3320 2
: 2047      3321 2 LOCAL
: 2048      3322 2      BMF : REF BLOCK [0, BYTE] FIELD (BSF$MAJOR_FRAME);
: 2049      3323 2
: 2050      3324 2 ++
: 2051      3325 2      Set the current data pointer in the frame to BUF_PTR + 1.
: 2052      3326 2      The one is added because Input initialize will subtract one from BUF_PTR.
: 2053      3327 2      This whole matter is explained in IO_BEG.
: 2054      3328 2 --
: 2055      3329 2
: 2056      3330 2 BMF = .CCB [ISB$A MAJ_F_PTR];
: 2057      3331 2 BMF [BSF$A_CUR_DTA] = -.CCB [LUB$A_BUF_PTR] + 1;
: 2058      3332 2 RETURN;
: 2059      3333 1 END;      ! End of routine BAS$$REC_RMF9
```

```
          50      FF48  CB  D0 00000 BAS$$REC_RMF9::
0087  C0      B0  AB          01  C1 00005      MOVL      -184(CCB), BMF      : 3330
          05 0000C      ADDL3  #1, -80(CCB), 135(BMF)      : 3331
          RSB      : 3333
```

; Routine Size: 13 bytes, Routine Base: \_BAS\$CODE + 05FE

BASS\$REC\_PROC  
1-095

: 2060

3334 1

J 8  
16-Sep-1984 01:01:12  
14-Sep-1984 11:56:35

VAX-11 Bliss-32 V4.0-742  
[BASRTL.SRC]BASRECPRO.B32;1

Page 56  
(23)



```
2062 3335 1 GLOBAL ROUTINE BAS$REC_GSE (                ! GET (sequential) a record
2063 3336 1     FOREIGN_BUFFER,
2064 3337 1     LOCK_FLAGS
2065 3338 1 ) : JSB_DO_READ NOVALUE =
2066 3339 1
2067 3340 1 !++
2068 3341 1 FUNCTIONAL DESCRIPTION:
2069 3342 1
2070 3343 1     Read one record. Update RECOUNT if successful.
2071 3344 1     If a foreign buffer is specified, then change RAB$L_RBF to point to the
2072 3345 1     "foreign buffer". Otherwise, signal a fatal error.
2073 3346 1
2074 3347 1 FORMAL PARAMETERS:
2075 3348 1
2076 3349 1     FOREIGN_BUFFER.rl.v           points to CB of foreign buffer or 0
2077 3350 1     LOCK_FLAGS.rlu.v             bits to set in RAB ROP for manual
2078 3351 1                               record locking (0 if none)
2079 3352 1 IMPLICIT INPUTS:
2080 3353 1
2081 3354 1     RAB$W_USZ                     User buffer size of foreign buffer
2082 3355 1     RAB$L_UBF                     Pointer to user buffer for foreign buffer
2083 3356 1     LUB$L_WAIT_TIME              Wait time for next input
2084 3357 1     WAIT                          The module level OWN WAIT
2085 3358 1
2086 3359 1 IMPLICIT OUTPUTS:
2087 3360 1
2088 3361 1     RAB$B_RAC                     Record access field
2089 3362 1     RECOUNT                     Own storage for RECOUNT function.
2090 3363 1     RAB$L_RBF                     Record pointer in RAB.
2091 3364 1     RAB$W_RSZ                     Number of bytes read (stored in RECOUNT)
2092 3365 1
2093 3366 1 ROUTINE VALUE:
2094 3367 1
2095 3368 1     NONE
2096 3369 1
2097 3370 1 SIDE EFFECTS:
2098 3371 1
2099 3372 1     RAB$W_USZ and RAB$W_UBF are altered while this routine is running,
2100 3373 1     but are restored before exit.
2101 3374 1     Reads next record from file on this logical unit.
2102 3375 1     SIGNALs any RMS errors
2103 3376 1 --
2104 3377 1
2105 3378 2 BEGIN
2106 3379 2
2107 3380 2 EXTERNAL REGISTER
2108 3381 2     CCB : REF BLOCK [, BYTE];
2109 3382 2
2110 3383 2 MAP
2111 3384 2     FOREIGN_BUFFER : REF BLOCK [, BYTE];
2112 3385 2
2113 3386 2 LOCAL
2114 3387 2     RMS_STATUS,
2115 3388 2     SAVE_USZ,                ! Save the USZ
2116 3389 2     ACTUAL_RSZ,              ! Actual record size
2117 3390 2     WAIT_TIME;              ! Current wait time
2118 3391 2 !+
```

```
2119 3392 2 1 Save USZ in case it is modified. It is faster to always
2120 3393 2 1 save and restore it, since that eliminates the test for foreign
2121 3394 2 1 buffers and single-character mode at the end of this routine.
2122 3395 2 1 -
2123 3396 2 1     SAVE_USZ = .CCB [RAB$W_USZ];
2124 3397 2 1 +
2125 3398 2 1 If a timeout has been specified, store information in the RAB to tell
2126 3399 2 1 RMS about it. If no timeout has been specified, clear the TMO bit
2127 3400 2 1 in case there was an earlier timeout specified.
2128 3401 2 1 -
2129 3402 2 1 +
2130 3403 2 1 If WAIT is zero then use the LUB's wait. This is to provide upward compatibility
2131 3404 2 1 , i.e. existing EXE's can run with the LUB wait value in V2.2.
2132 3405 2 1 -
2133 3406 2 1     WAIT_TIME = ( IF ( .WAIT EQL 0 ) THEN .CCB [ LUB$L_WAIT_TIME ] ELSE .WAIT );
2134 3407 2 1
2135 3408 2 1     IF ( .WAIT_TIME EQL 0 )
2136 3409 2 1     THEN
2137 3410 2 1         CCB [RAB$V_TMO] = 0
2138 3411 2 1     ELSE
2139 3412 2 1         BEGIN
2140 3413 2 1             CCB [RAB$B_TMO] = .WAIT_TIME;
2141 3414 2 1             CCB [RAB$V_TMO] = 1;
2142 3415 2 1         END;
2143 3416 2 1
2144 3417 2 1 +
2145 3418 2 1 Set the Read-no-echo RMS bit based on the user's last call to
2146 3419 2 1 ECHO or NOECHO.
2147 3420 2 1 -
2148 3421 2 1     CCB [RAB$V_RNE] = .CCB [LUB$V_NOECHO];
2149 3422 2 1 +
2150 3423 2 1 Set the record pointer field in the RAB to the user buffer. This is
2151 3424 2 1 done on each element transmission and not just at OPEN because of RMS
2152 3425 2 1 Locate mode.
2153 3426 2 1 Fill the input buffer with Nulls. Basic semantics require null fill if
2154 3427 2 1 the record read is shorter than the buffer.
2155 3428 2 1 Set the record access field in the RAB to sequential. Perform the GET.
2156 3429 2 1 If RMS returns a failure status, signal the error. If the GET is
2157 3430 2 1 successful, then update recount.
2158 3431 2 1 -
2159 3432 2 1     IF ( .FOREIGN_BUFFER NEQA 0 )
2160 3433 2 1     THEN
2161 3434 2 1         BEGIN
2162 3435 2 1 +
2163 3436 2 1 A foreign buffer was specified. Save off RAB$L_usz of the "file" channel
2164 3437 2 1 and then substitute the appropriate values from the foreign channel into
2165 3438 2 1 the file channel to make the record go directly into the foreign buffer.
2166 3439 2 1 -
2167 3440 2 1     CCB [RAB$W_USZ] = .FOREIGN_BUFFER [RAB$W_USZ];
2168 3441 2 1     END;
2169 3442 2 1
2170 3443 2 1 +
2171 3444 2 1 If the user has called ONECHR, shrink the effective size of the
2172 3445 2 1 buffer to one character, so he will get characters one at a time.
2173 3446 2 1 -
2174 3447 2 1
2175 3448 2 1
```



```
2176 3449 2 | The user must call ONECHR before each GET, so we clear the ONECHR
2177 3450 2 | flag here.
2178 3451 2 | -
2179 3452 2 |
2180 3453 2 | IF (.CCB [LUB$V_ONECHR])
2181 3454 2 | THEN
2182 3455 2 | BEGIN
2183 3456 2 | CCB [LUB$V_ONECHR] = 0;
2184 3457 2 | CCB [RAB$W_USZ] = 1;
2185 3458 2 | END;
2186 3459 2 |
2187 3460 2 | CCB [RAB$B_RAC] = RAB$C_SEQ;
2188 3461 2 |
2189 3462 2 | +
2190 3463 2 | Set bits in the RAB ROP (careful not to turn off ULK).
2191 3464 2 | -
2192 3465 2 |
2193 3466 2 | CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] OR .LOCK_FLAGS;
2194 3467 2 |
2195 3468 2 | RMS_STATUS = $GET (RAB = .CCB);
2196 3469 2 |
2197 3470 2 | IF .RMS_STATUS EQL RMS$_CTRLC
2198 3471 2 | THEN
2199 3472 2 | BAS$$SIGNAL_CTRLC ();
2200 3473 2 |
2201 3474 2 | IF NOT .RMS_STATUS
2202 3475 2 | THEN
2203 3476 2 | BEGIN
2204 3477 2 | +
2205 3478 2 | We cannot call GET_ERROR because we must restore UBF and USZ.
2206 3479 2 | -
2207 3480 2 |
2208 3481 2 | WHILE (.CCB [RAB$L_STS] EQL RMS$_RSA) DO
2209 3482 2 | BEGIN
2210 3483 2 | $WAIT (RAB = .CCB);
2211 3484 2 | $GET (RAB = .CCB);
2212 3485 2 | END;
2213 3486 2 |
2214 3487 2 | END;
2215 3488 2 |
2216 3489 2 | +
2217 3490 2 | Clear RAB ROP bits so that a subsequent I/O operation does not
2218 3491 2 | inherit them.
2219 3492 2 | -
2220 3493 2 |
2221 3494 2 | CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] XOR .LOCK_FLAGS;
2222 3495 2 |
2223 3496 2 | +
2224 3497 2 | This actual record size may or may not change below. If the file is a
2225 3498 2 | terminal device then it will get terminators tacked on to the record read.
2226 3499 2 | -
2227 3500 2 |
2228 3501 2 | ACTUAL_RSZ = .CCB [RAB$W_RSZ];
2229 3502 2 |
2230 3503 2 | +
2231 3504 2 | Tack on the terminators when a terminal device file, just like INPUT LINE
2232 3505 2 | -
```

```
2233 3506 2 IF .CCB[LUB$V_TERM_DEV]
2234 3507 2 THEN
2235 3508 2 BEGIN
2236 3509 2 LITERAL K_ESCAPE = %X'1B',
2237 3510 2 K_CR = %X'0D',
2238 3511 2 K_CRLF = %X'0A0D';
2239 3512 2
2240 3513 2 !+ STV0 is the escape character, STV2 is the number of terminating characters.
2241 3514 2 !-
2242 3515 2 SELECTONEU .CCB [RAB$W_STV0] OF
2243 3516 2 SET
2244 3517 2 [K_ESCAPE] :
2245 3518 2 BEGIN
2246 3519 2 !+
2247 3520 2 ! If the length is one then escape is not at the end of the buffer and it
2248 3521 2 ! must be moved there, otherwise the escape sequence is at the end of the
2249 3522 2 ! buffer following the data.
2250 3523 2
2251 3524 2 IF .CCB [RAB$W_STV2] EQLU 1
2252 3525 2 THEN
2253 3526 2 BEGIN
2254 3527 2 IF .CCB [RAB$W_RSZ] GEQU .CCB [RAB$W_USZ]
2255 3528 2 THEN BAS$$STOP_IO (BAS$K_RECFLT00);
2256 3529 2 CH$MOVE (1,UPLIT(K_ESCAPE),.CCB [RAB$L_UBF] + .CCB [RAB$W_RSZ]);
2257 3530 2 ACTUAL_RSZ = .ACTUAL_RSZ + 1;
2258 3531 2 END
2259 3532 2 ELSE
2260 3533 2 ACTUAL_RSZ = .ACTUAL_RSZ + .CCB [RAB$W_STV2];
2261 3534 2 END;
2262 3535 2 [K_CR] :
2263 3536 2 BEGIN
2264 3537 2 IF .CCB [RAB$W_RSZ] + 2 GTRU .CCB [RAB$W_USZ]
2265 3538 2 THEN BAS$$STOP_IO (BAS$K_RECFLT00);
2266 3539 2 CH$MOVE (2,UPLIT(K_CRLF),.CCB [RAB$L_UBF] + .CCB [RAB$W_RSZ]);
2267 3540 2 ACTUAL_RSZ = .ACTUAL_RSZ + 2;
2268 3541 2 END;
2269 3542 2 [OTHERWISE] :
2270 3543 2 ;
2271 3544 2 ;
2272 3545 2 TES;
2273 3546 2 END;
2274 3547 2
2275 3548 2 !+
2276 3549 2 ! If there are no errors, null pad the buffer.
2277 3550 2 !-
2278 3551 2
2279 3552 2 IF (.CCB [RAB$W_USZ] GTR .ACTUAL_RSZ) AND .CCB [RAB$L_STS]
2280 3553 2 THEN
2281 3554 2 CH$FILL (%X'00',
2282 3555 2 .CCB [RAB$W_USZ] - .ACTUAL_RSZ, .CCB [RAB$L_UBF] + .ACTUAL_RSZ);
2283 3556 2
2284 3557 2 !+
2285 3558 2 ! Before checking for errors, restore UBF and USZ, and set RECOUNT.
2286 3559 2 !-
2287 3560 2 CCB [RAB$L_UBF] = .CCB [LUB$A_UBF];
2288 3561 2 CCB [RAB$W_USZ] = .SAVE_USZ;
2289 3562 2 RECOUNT = .ACTUAL_RSZ;
```



```
: 2290      3563 2 1+
: 2291      3564 2 1+ Any error remaining (which will be an error other than Record Stream
: 2292      3565 2 1+ Active, RSA) is fatal.
: 2293      3566 2 1+
: 2294      3567 2 1+
: 2295      3568 2 1+
: 2296      3569 2 1+
: 2297      3570 2 1+
: 2298      3571 2 1+
: 2299      3572 1 1+
      IF ( NOT .CCB [RAB$L_STS]) THEN BAS$$STOP_IO (BAS$K_IOERR_REC);
      CCB [LUB$A_RBUF_ADR] = .CCB [RAB$L_RBF];
      RETURN;
      END;
      ! End of BAS$$REC_GSE
```

```
0000001B 0060B .BLKB 1
00000A0D 0060C P.AAA: .LONG 27
00000A0D 00610 P.AAB: .LONG 2573
      .EXTRN SYSS$WAIT
3C BB 00000 BAS$$REC_GSE::
      SE 10 C2 00002 PUSHR #^M<R2,R3,R4,R5>
      54 51 D0 00005 SUBL2 #16, SP
      08 AE 20 AB 9E 00008 MOVL R1, R4
      04 AE 08 BE 3C 0000D MOVAB 32(CCB), 8(SP)
      51 00000000 EF 3C 00012 MOVZWL @8(SP), SAVE_USZ
      04 12 00019 MOVZWL WAIT, R1
      51 CC AB D0 0001B BNEQ 1$
      52 04 AB 9E 0001F 1$: MOVL -52(CCB), WAIT_TIME
      51 D5 00023 MOVAB 4(CCB), R2
      06 12 00025 TSTL WAIT_TIME
      03 A2 02 8A 00027 BNEQ 2$
      08 11 0002B BICB2 #2, 3(R2)
      1F AB 51 90 0002D 2$: BRB 3$
      03 A2 02 88 00031 MOVB WAIT_TIME, 31(CCB)
      00 A0 AB F0 00035 3$: BISB2 #2, 3(R2)
      50 D5 0003C INSV -96(CCB), #0, #1, 3(R2)
      05 13 0003E TSTL FOREIGN_BUFFER
      08 BE 20 A0 B0 00040 BEQL 4$
      A0 AB 01 E1 00045 4$: MOVW 32(Foreign Buffer), @8(SP)
      A0 AB 02 8A 0004A BBC #1, -96(CCB), 5$
      08 BE 01 B0 0004E BICB2 #2, -96(CCB)
      62 1E AB 94 00052 5$: MOVW #1, @8(SP)
      54 C8 00055 CLRB 30(CCB)
      5B DD 00058 BISL2 LOCK_FLAGS, (R2)
      01 FB 0005A PUSHL CCB
      50 D0 00061 CALLS #1, SYSS$GET
      53 D1 00064 MOVL R0, RMS_STATUS
      07 12 0006B CMPL RMS_STATUS, #67153
      00 FB 0006D BNEQ 6$
      53 E8 00074 6$: CALLS #0, BAS$$SIGNAL_CTRL_C
      08 AB D1 00077 7$: BLBS RMS_STATUS, 8$
      14 12 0007F CMPL 8(CCB), #99034
      5B DD 00081 BNEQ 8$
      01 FB 00083 PUSHL CCB
      5B DD 0008A PUSHL CCB
      00 01 FB 0008C CALLS #1, SYSS$WAIT
      00 01 FB 0008C CALLS #1, SYSS$GET
```



: 2300 3573 1



```
2302 3574 1 GLOBAL ROUTINE BASS$REC_GIN ( ! GET (indexed) a record
2303 3575 1 KEY_NO, REL_OP, KEY, FOREIGN_BUFFER, LOCK_FLAGS) : JSB_REC_IND1 NOVALUE =
2304 3576 1
2305 3577 1 ++
2306 3578 1 FUNCTIONAL DESCRIPTION:
2307 3579 1
2308 3580 1 Read one record. Update RECOUNT if successful.
2309 3581 1 If a foreign buffer is specified, then change RAB$RBF to point to the
2310 3582 1 "foreign buffer". Otherwise, signal a fatal error.
2311 3583 1
2312 3584 1 FORMAL PARAMETERS:
2313 3585 1
2314 3586 1 KEY_NO.rl.v key of reference number
2315 3587 1 REL_OP.rl.v relative relationship of the key
2316 3588 1 KEY.rt.dx key to search for
2317 3589 1 FOREIGN_BUFFER.rl.v points to CB of foreign buffer or 0
2318 3590 1 LOCK_FLAGS.rlu.v bits to set in RAB ROP to control manual record
2319 3591 1 locking (0 if none)
2320 3592 1
2321 3593 1 IMPLICIT INPUTS:
2322 3594 1
2323 3595 1 RAB$W_USZ User buffer size of foreign buffer
2324 3596 1 RAB$L_UBF Pointer to user buffer for foreign buffer
2325 3597 1
2326 3598 1 IMPLICIT OUTPUTS:
2327 3599 1
2328 3600 1 RAB$B_RAC Record access field
2329 3601 1 RECOUNT Own storage for RECOUNT function.
2330 3602 1 RAB$L_RBF Record pointer in RAB.
2331 3603 1 RAB$W_USZ User buffer size for "file" buffer
2332 3604 1 RAB$L_UBF Pointer to user buffer for "file" buffer
2333 3605 1
2334 3606 1 ROUTINE VALUE:
2335 3607 1
2336 3608 1 NONE
2337 3609 1
2338 3610 1 SIDE EFFECTS:
2339 3611 1
2340 3612 1 Reads next record from file on this logical unit.
2341 3613 1 SIGNALS any RMS errors
2342 3614 1 --
2343 3615 1
2344 3616 2 BEGIN
2345 3617 2
2346 3618 2 EXTERNAL REGISTER
2347 3619 2 CCB : REF BLOCK [, BYTE];
2348 3620 2
2349 3621 2 MAP
2350 3622 2 KEY : REF BLOCK [8, BYTE], ! descriptor of the key
2351 3623 2 FOREIGN_BUFFER : REF BLOCK [, BYTE];
2352 3624 2
2353 3625 2 LITERAL
2354 3626 2 K_EQUAL = 0, ! keys should be equal
2355 3627 2 K_GREATER_EQUAL = 1, ! key should be greater than or equal
2356 3628 2 K_GREATER_THAN = 2; ! key should be greater than
2357 3629 2
2358 3630 2 LOCAL
```

```
2359 3631 2 RMS STATUS,
2360 3632 2 SAVE_USZ;
2361 3633 2 ! Save the USZ
2362 3634 2
2363 3635 2 + Save USZ in case it is modified. It is faster to always
2364 3636 2 save and restore it, since that eliminates the test for foreign
2365 3637 2 buffers and single-character mode at the end of this routine.
2366 3638 2 -
2367 3639 2 SAVE_USZ = .CCB [RAB$W_USZ];
2368 3640 2
2369 3641 2 +
2370 3642 2 Set the record pointer field in the RAB to the user buffer. This is
2371 3643 2 done on each element transmission and not just at OPEN because of RMS
2372 3644 2 Locate mode.
2373 3645 2 Fill the input buffer with Nulls. Basic semantics require null fill if
2374 3646 2 the record read is shorter than the buffer.
2375 3647 2 Set the record access field in the RAB to sequential. Perform the GET.
2376 3648 2 If RMS returns a failure status, signal the error. If the GET is
2377 3649 2 successful, then update recount.
2378 3650 2 -
2379 3651 2
2380 3652 2 IF .FOREIGN_BUFFER NEQ 0
2381 3653 2 THEN
2382 3654 2 BEGIN
2383 3655 2 +
2384 3656 2 A foreign buffer was specified. Substitute the appropriate values from the foreign channel into
2385 3657 2 the file channel to make the record go directly into the foreign buffer.
2386 3658 2 -
2387 3659 2 CCB [RAB$L_UBF] = .FOREIGN_BUFFER [RAB$L_UBF];
2388 3660 2 CCB [RAB$W_USZ] = .FOREIGN_BUFFER [RAB$W_USZ];
2389 3661 2 END;
2390 3662 2
2391 3663 2 +
2392 3664 2 Set the record access field to key. Set KBF to the key. Set KSZ to the
2393 3665 2 the size of the key passed. Set the key of reference to the desired key.
2394 3666 2 Use a case statement to toggle KGT and KGE in the ROP.
2395 3667 2 -
2396 3668 2
2397 3669 2 CCB [RAB$B_RAC] = RAB$C_KEY;
2398 3670 2 CCB [RAB$L_KBF] = .KEY [DSC$A_POINTER];
2399 3671 2 CCB [RAB$B_KRF] = .KEY_NO;
2400 3672 2 CCB [RAB$B_KSZ] = (IF .KEY [DSC$B_DTYPE] NEQ DSC$K_DTYPE_P
2401 3673 2 THEN
2402 3674 2 .KEY [DSC$W_LENGTH]
2403 3675 2 ELSE
2404 3676 2 (.KEY [DSC$W_LENGTH]/2 + 1));
2405 3677 2
2406 3678 2 CASE .REL_OP FROM K_EQUAL TO K_GREATER_THAN OF
2407 3679 2 SET
2408 3680 2 [K_EQUAL] :
2409 3681 2 CCB [RAB$V_KGE] = CCB [RAB$V_KGT] = 0;
2410 3682 2
2411 3683 2 [K_GREATER_EQUAL] :
2412 3684 2 BEGIN
2413 3685 2 CCB [RAB$V_KGE] = 1;
2414 3686 2 CCB [RAB$V_KGT] = 0;
2415 3687 2
```



```
2416 3688 2      END;
2417 3689 2
2418 3690 2      [K_GREATER_THAN] :
2419 3691 2      BEGIN
2420 3692 2          CCB [RAB$V_KGT] = 1;
2421 3693 2          CCB [RAB$V_KGE] = 0;
2422 3694 2      END;
2423 3695 2      TES;
2424 3696 2
2425 3697 2      + Set bits in RAB ROP without turning off ULK.
2426 3698 2      -
2427 3699 2
2428 3700 2          CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] OR .LOCK_FLAGS;
2429 3701 2
2430 3702 2          RMS_STATUS = $GET (RAB = .CCB);
2431 3703 2
2432 3704 2          IF .RMS_STATUS EQL RMSS_CONTROLC
2433 3705 2          THEN
2434 3706 2              BAS$$SIGNAL_CTRLC ();
2435 3707 2
2436 3708 2          IF NOT .RMS_STATUS
2437 3709 2          THEN
2438 3710 2              BEGIN
2439 3711 2
2440 3712 2      + We cannot call GET_ERROR because we must restore UBF and USZ.
2441 3713 2      -
2442 3714 2
2443 3715 2          WHILE (.CCB [RAB$L_STS] EQL RMSS_RSA) DO
2444 3716 2              BEGIN
2445 3717 2                  $WAIT (RAB = .CCB);
2446 3718 2                  $GET (RAB = .CCB);
2447 3719 2                  END;
2448 3720 2
2449 3721 2              END;
2450 3722 2
2451 3723 2      + Turn off bits in RAB ROP so that subsequent I/O operations can not
2452 3724 2      - inherit them.
2453 3725 2
2454 3726 2          CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] XOR .LOCK_FLAGS;
2455 3727 2
2456 3728 2
2457 3729 2      + If there are no errors, null pad the buffer.
2458 3730 2      -
2459 3731 2
2460 3732 2          IF (.CCB [RAB$W_USZ] GTR .CCB [RAB$W_RSZ]) AND .CCB [RAB$L_STS]
2461 3733 2          THEN
2462 3734 2              CH$FILL ('X'00',
2463 3735 2                  .CCB [RAB$W_USZ] - .CCB [RAB$W_RSZ], .CCB [RAB$L_UBF] + .CCB [RAB$W_RSZ]);
2464 3736 2
2465 3737 2
2466 3738 2      + Before checking for errors, restore UBF and USZ, and set RECOUNT.
2467 3739 2      -
2468 3740 2
2469 3741 2          CCB [RAB$L_UBF] = .CCB [LUB$A_UBF];
2470 3742 2          CCB [RAB$W_USZ] = .SAVE_USZ;
2471 3743 2
2472 3744 2
```

```

: 2473      3745 2   RECOUNT = .CCB [RAB$W_RSZ];
: 2474      3746 2   +
: 2475      3747 2   Any error remaining (which will be an error other than Record Stream
: 2476      3748 2   Active, RSA) is fatal.
: 2477      3749 2   -
: 2478      3750 2
: 2479      3751 2   IF ( NOT .CCB [RAB$L_STS]) THEN BAS$$STOP_IO (BAS$K_IOERR_REC);
: 2480      3752 2
: 2481      3753 2   +
: 2482      3754 2   This is frosting on the cake. LUB$A RBUF_ADR points to the record buffer for
: 2483      3755 2   MOVE. The buffer may change due to RMS Locate Mode. Currently, Locate Mode
: 2484      3756 2   is not performed on files which UPDATE or PUT. However, in anticipation that
: 2485      3757 2   RMS may add such a capability, we point RBUF_ADR off to the buffer used by PUT.
: 2486      3758 2   -
: 2487      3759 2   CCB [LUB$A_RBUF_ADR] = .CCB [RAB$L_RBF];
: 2488      3760 2   RETURN;
: 2489      3761 1   END;

```

! End of BAS\$\$REC\_GIN

			3C	BB	00000	BAS\$\$REC_GIN::			
						PUSHR	#^M<R2,R3,R4,R5>	3574	
	7E	20	AB	3C	00002	MOVZWL	32(CCB), SAVE USZ	3639	
			53	D5	00006	TSTL	FOREIGN_BUFFER	3652	
			0A	13	00008	BEQL	1\$		
24	AB	24	A3	D0	0000A	MOVL	36(FOREIGN_BUFFER), 36(CCB)	3659	
20	AB	20	A3	B0	0000F	MOVW	32(FOREIGN_BUFFER), 32(CCB)	3660	
1E	AB		01	90	00014	1\$:	MOVW	3669	
30	AB	04	A2	D0	00018	MOVW	#1, 30(CCB)	3670	
35	AB		50	90	0001D	MOVW	4(KEY), 48(CCB)	3671	
	15	02	A2	91	00021	MOVW	KEY NO, 53(CCB)	3672	
			05	13	00025	CMPB	2(KEY), #21		
	52		62	3C	00027	BEQL	2\$		
			08	11	0002A	MOVZWL	(KEY), R2	3674	
	52		62	3C	0002C	BRB	3\$		
	52		02	C6	0002F	MOVZWL	(KEY), R2	3676	
			52	D6	00032	DIVL2	#2, R2		
34	AB		52	90	00034	INCL	R2		
	52	04	AB	9E	00038	MOVW	R2, 52(CCB)	3672	
02	00		51	CF	0003C	MOVAB	4(CCB), R2	3682	
0018	000D		0006	00040	4\$:	CASEL	REL OP, #0, #2	3678	
						.WORD	5\$-4\$,-		
							6\$-4\$,-		
							7\$-4\$,-		
02	A2	40	8F	8A	00046	5\$:	BICB2	#64, 2(R2)	3682
			10	11	0004B		BRB	8\$	
02	A2		20	88	0004D	6\$:	BISB2	#32, 2(R2)	3686
02	A2	40	8F	8A	00051		BICB2	#64, 2(R2)	3687
			09	11	00056		BRB	9\$	3678
02	A2	40	8F	88	00058	7\$:	BISB2	#64, 2(R2)	3692
02	A2		20	8A	0005D	8\$:	BICB2	#32, 2(R2)	3693
	62		54	C8	00061	9\$:	BISL2	LOCK_FLAGS, (R2)	3701
			5B	DD	00064		PUSHL	CCB	3703
00000000G	00		01	FB	00066		CALLS	#1, SYSSGET	
	53		50	D0	0006D		MOVL	R0, RMS STATUS	
00010651	8F		53	D1	00070		CMPL	RMS_STATUS, #67153	3705



			07	12	00077	BNEQ	10\$		
			00	FB	00079	CALLS	#0, BAS\$\$SIGNAL_CTRL		3707
			53	E8	00080	BLBS	RMS STATUS, 12\$		3709
			AB	D1	00083	CMPL	8(CCB), #99034		3716
			14	12	00088	BNEQ	12\$		
			5B	DD	0008D	PUSHL	CCB		3718
			01	FB	0008F	CALLS	#1, SYSSWAIT		
			5B	DD	00096	PUSHL	CCB		3719
			01	FB	00098	CALLS	#1, SYSSGET		
			E2	11	0009F	BRB	11\$		3716
			54	CC	000A1	XORL2	LOCK_FLAGS, (R2)		3729
			AB	B1	000A4	CMPL	32(CCB), 34(CCB)		3735
			1D	1B	000A9	BLEQU	13\$		
			AB	E9	000AB	BLBC	8(CCB), 13\$		
			AB	3C	000AF	MOVZWL	32(CCB), R1		3738
			AB	3C	000B3	MOVZWL	34(CCB), R0		
			50	C2	000B7	SUBL2	R0, R1		
			AB	3C	000BA	MOVZWL	34(CCB), R0		
			AB	C0	000BE	ADDL2	36(CCB), R0		
			00	2C	000C2	MOVCS	#0, (SP), #0, R1, (R0)		
			60		000C7				
			AB	D0	000C8	MOVL	-100(CCB), 36(CCB)		3743
			6E	B0	000CD	MOVW	SAVE_USZ, 32(CCB)		3744
			AB	3C	000D1	MOVZWL	34(CCB), RECOUNT		3745
			AB	E8	000D9	BLBS	8(CCB), 14\$		3751
			01	CE	000DD	MNEGL	#1, -(SP)		
			01	FB	000E0	CALLS	#1, BAS\$\$STOP_10		
			AB	D0	000E7	MOVL	40(CCB), -20(CCB)		3759
			04	C0	000EC	ADDL2	#4, SP		3761
			3C	BA	000EF	POPR	#M<R2,R3,R4,R5>		
			05		000F1	RSB			

; Routine Size: 242 bytes, Routine Base: \_BAS\$CODE + 0766

; 2490 3762 1

```
2492 3763 1 GLOBAL ROUTINE BAS$$REC_GRE ( ! GET (relative) a record
2493 3764 1 FOREIGN_BUFFER, LOCK_FLAGS) : JSB_DO_READ NOVALUE =
2494 3765 1
2495 3766 1 !++
2496 3767 1 FUNCTIONAL DESCRIPTION:
2497 3768 1
2498 3769 1 Read one record from a relative file. Modify the RAB if necessary for
2499 3770 1 foreign buffers. Update RECOUNT if successful. Otherwise, signal a fatal error.
2500 3771 1
2501 3772 1 FORMAL PARAMETERS:
2502 3773 1
2503 3774 1 FOREIGN_BUFFER.rl.v The address of the CB of a foreign
2504 3775 1 buffer or 0
2505 3776 1 LOCK_FLAGS.rlu.v bits to set in the RAB ROP to control
2506 3777 1 manual record locking (0 if none)
2507 3778 1 IMPLICIT INPUTS:
2508 3779 1
2509 3780 1 RAB$W_RSZ record size for foreign buffer
2510 3781 1 RAB$L_UBF buffer address for foreign buffer
2511 3782 1
2512 3783 1 IMPLICIT OUTPUTS:
2513 3784 1
2514 3785 1 RAB$B_RAC Record access field
2515 3786 1 RECOUNT Own storage for RECOUNT function.
2516 3787 1 RAB$L_RBF Record pointer in RAB.
2517 3788 1 RAB$W_USZ record size of file buffer
2518 3789 1 RAB$L_UBF address of buffer for file buffer
2519 3790 1
2520 3791 1 ROUTINE VALUE:
2521 3792 1
2522 3793 1 NONE
2523 3794 1
2524 3795 1 SIDE EFFECTS:
2525 3796 1
2526 3797 1 -- SIGNALS any RMS errors
2527 3798 1
2528 3799 1
2529 3800 2 BEGIN
2530 3801 2
2531 3802 2 EXTERNAL REGISTER
2532 3803 2 CCB : REF BLOCK [, BYTE];
2533 3804 2
2534 3805 2 MAP
2535 3806 2 FOREIGN_BUFFER : REF BLOCK [, BYTE];
2536 3807 2
2537 3808 2 LOCAL
2538 3809 2 RMS_STATUS,
2539 3810 2 SAVE_USZ; ! Save the USZ
2540 3811 2
2541 3812 2 !+
2542 3813 2 Save USZ in case it is modified. It is faster to always
2543 3814 2 save and restore it, since that eliminates the test for foreign
2544 3815 2 buffers and single-character mode at the end of this routine.
2545 3816 2 --
2546 3817 2 SAVE_USZ = .CCB [RAB$W_USZ];
2547 3818 2
2548 3819 2 !+
```



```
2549 3820 2  | Set the record pointer field in the RAB to the user buffer. This is
2550 3821 2  | done on each element transmission and not just at OPEN because of RMS
2551 3822 2  | Locate mode.
2552 3823 2  | Fill the input buffer with Nulls. Basic semantics require null fill if
2553 3824 2  | the record read is shorter than the buffer.
2554 3825 2  | Set the record access field in the RAB to sequential. Perform the GET.
2555 3826 2  | If RMS returns a failure status, signal the error. If the GET is
2556 3827 2  | successful, then update recount.
2557 3828 2  |
2558 3829 2  |
2559 3830 2  | IF .FOREIGN_BUFFER NEQ 0
2560 3831 2  | THEN
2561 3832 2  | BEGIN
2562 3833 2  | +
2563 3834 2  | There is a foreign buffer. Modify the file buffer to point to the
2564 3835 2  | buffer associated with the foreign buffer's channel.
2565 3836 2  | -
2566 3837 2  | CCB [RAB$L_RBF] = CCB [RAB$L_UBF] = .FOREIGN_BUFFER [RAB$L_UBF];
2567 3838 2  | CCB [RAB$W_RSZ] = CCB [RAB$W_USZ] = .FOREIGN_BUFFER [RAB$W_USZ];
2568 3839 2  | END
2569 3840 2  | ELSE
2570 3841 2  | CCB [RAB$L_RBF] = .CCB [RAB$L_UBF];
2571 3842 2  |
2572 3843 2  | CCB [RAB$B_RAC] = RAB$C_KEY;
2573 3844 2  |
2574 3845 2  | +
2575 3846 2  | Set bits in RAB ROP without destroying ULK.
2576 3847 2  | -
2577 3848 2  |
2578 3849 2  | CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] OR .LOCK_FLAGS;
2579 3850 2  |
2580 3851 2  | RMS_STATUS = $GET (RAB = .CCB);
2581 3852 2  |
2582 3853 2  | IF .RMS_STATUS EQL RMSS_CONTROLC
2583 3854 2  | THEN
2584 3855 2  | BAS$$SIGNAL_CTRLC ();
2585 3856 2  |
2586 3857 2  | IF NOT .RMS_STATUS
2587 3858 2  | THEN
2588 3859 2  | BEGIN
2589 3860 2  | +
2590 3861 2  | We cannot call GET_ERROR because we must restore UBF and USZ.
2591 3862 2  | -
2592 3863 2  |
2593 3864 2  | WHILE (.CCB [RAB$L_STS] EQL RMSS_RSA) DO
2594 3865 2  | BEGIN
2595 3866 2  | $WAIT (RAB = .CCB);
2596 3867 2  | $GET (RAB = .CCB);
2597 3868 2  | END;
2598 3869 2  |
2599 3870 2  | END;
2600 3871 2  |
2601 3872 2  | +
2602 3873 2  | Turn off bits in the RAB ROP so that subsequent I/O operations can not
2603 3874 2  | inherit them.
2604 3875 2  | -
2605 3876 2  |
```

```
2606 3877 2 CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] XOR .LOCK_FLAGS;
2607 3878
2608 3879 +
2609 3880 - If there are no errors, null pad the buffer.
2610 3881
2611 3882
2612 3883 IF (.CCB [RAB$W_USZ] GTR .CCB [RAB$W_RSZ]) AND .CCB [RAB$L_STS]
2613 3884 THEN
2614 3885 CH$FILL (%X'00',
2615 3886 .CCB [RAB$W_USZ] - .CCB [RAB$W_RSZ], .CCB [RAB$L_UBF] + .CCB [RAB$W_RSZ]);
2616 3887
2617 3888 +
2618 3889 - Before checking for errors, restore UBF and USZ, and set RECOUNT.
2619 3890
2620 3891 CCB [RAB$L_UBF] = .CCB [LUB$A_UBF];
2621 3892 CCB [RAB$W_USZ] = .SAVE_USZ;
2622 3893 RECOUNT = .CCB [RAB$W_RSZ];
2623 3894
2624 3895 +
2625 3896 - Any error remaining (which will be an error other than Record Stream
2626 3897 Active, RSA) is fatal.
2627 3898
2628 3899 IF ( NOT .CCB [RAB$L_STS]) THEN BAS$$STOP_IO (BAS$K_IOERR_REC);
2629 3900
2630 3901 +
2631 3902 - Set LUB$A_RBUF_ADR to point to the buffer used by RMS. It may move around
2632 3903 due to Locate Mode.
2633 3904
2634 3905 CCB [LUB$A_RBUF_ADR] = .CCB [RAB$L_RBF];
2635 3906 RETURN;
2636 3907 1 END;
! End of BAS$$REC_GRE
```

			3C	BB	00000	BAS\$\$REC GRE::		
						PUSHR	#*M<R2,R3,R4,R5>	3763
	52		51	D0	00002	MOVL	R1, R2	
	7E	20	AB	3C	00005	MOVZWL	32(CCB), SAVE_USZ	3817
			50	D5	00009	TSTL	FOREIGN_BUFFER	3830
			1A	13	0000B	BEQL	1\$	
	51	24	A0	D0	0000D	MOVL	36(FOREIGN_BUFFER), R1	3837
24	AB		51	D0	00011	MOVL	R1, 36(CCB)	
28	AB		51	D0	00015	MOVL	R1, 40(CCB)	
	50	20	A0	3C	00019	MOVZWL	32(FOREIGN_BUFFER), R0	3838
20	AB		50	B0	0001D	MOVW	R0, 32(CCB)	
22	AB		50	B0	00021	MOVW	R0, 34(CCB)	
			05	11	00025	BRB	2\$	3830
28	AB	24	AB	D0	00027	MOVL	36(CCB), 40(CCB)	3841
1E	AB		01	90	0002C	MOVB	#1, 30(CCB)	3843
04	AB		52	C8	00030	BISL2	LOCK_FLAGS, 4(CCB)	3849
			5B	DD	00034	PUSHL	CCB	3851
00000000G	00		01	FB	00036	CALLS	#1, SYS\$GET	
	53		50	D0	0003D	MOVL	R0, RMS_STATUS	
00010651	8F		53	D1	00040	CMPL	RMS_STATUS, #67153	3853
			07	12	00047	BNEQ	3\$	



00000000G	00	00	FB	00049	CALLS	#0, BASS\$SIGNAL_CTRL	3855
	1E	53	E8	00050	BLBS	RMS STATUS, 5\$	3857
000182DA	8F	08	AB	D1 00053	CMPL	8(CCB), #99034	3864
			14	12 0005B	BNEQ	5\$	
			5B	DD 0005D	PUSHL	CCB	3866
00000000G	00		01	FB 0005F	CALLS	#1, SYSS\$WAIT	
			5B	DD 00066	PUSHL	CCB	3867
00000000G	00		01	FB 00068	CALLS	#1, SYSS\$GET	
			E2	11 0006F	BRB	4\$	3864
	04	AB	52	CC 00071	XORL2	LOCK FLAGS, 4(CCB)	3877
	22	AB	20	AB B1 00075	CMPW	32(CCB), 34(CCB)	3883
			1D	1B 0007A	BLEQU	6\$	
		19	08	AB E9 0007C	BLBC	8(CCB), 6\$	
		51	20	AB 3C 00080	MOVZWL	32(CCB), R1	3886
		50	22	AB 3C 00084	MOVZWL	34(CCB), R0	
		51	50	C2 00088	SUBL2	R0, R1	
		50	22	AB 3C 0008B	MOVZWL	34(CCB), R0	
		50	24	AB C0 0008F	ADDL2	36(CCB), R0	
51	00	6E	00	2C 00093	MOVCS	#0, (SP), #0, R1, (R0)	
			60	00098			
	24	AB	9C	AB D0 00099	MOVL	-100(CCB), 36(CCB)	3891
	20	AB	6E	B0 0009E	MOVW	SAVE USZ, 32(CCB)	3892
00000000'	EF	22	AB	3C 000A2	MOVZWL	34(CCB), RECOUNT	3893
	0A	08	AB	E8 000AA	BLBS	8(CCB), 7\$	3899
	7E		01	CE 000AE	MNEGL	#1, -(SP)	
00000000G	00		01	FB 000B1	CALLS	#1, BASS\$STOP IO	
	EC	28	AB	D0 000B8	MOVL	40(CCB), -20(CCB)	3905
			04	C0 000BD	ADDL2	#4, SP	3907
			3C	BA 000C0	POPR	#^M<R2,R3,R4,R5>	
			05	000C2	RSB		

; Routine Size: 195 bytes, Routine Base: \_BASS\$CODE + 0858

```
2638 3908 1 GLOBAL ROUTINE BASS$REC_GRFA ( ! GET (by RFA) a record
2639 3909 1 FOREIGN_BUFFER, LOCK_FLAGS) : JSB_DO_READ NOVALUE =
2640 3910 1
2641 3911 1 !++
2642 3912 1 FUNCTIONAL DESCRIPTION:
2643 3913 1
2644 3914 1 Read one record from a file. Modify the RAB if necessary for
2645 3915 1 foreign buffers. Update RECOUNT if successful. Otherwise, signal a fatal error.
2646 3916 1
2647 3917 1 FORMAL PARAMETERS:
2648 3918 1
2649 3919 1 FOREIGN_BUFFER.rl.v The address of the CB of a foreign
2650 3920 1 buffer or 0
2651 3921 1 LOCK_FLAGS.rlu.v bits to set in the RAB ROP to control
2652 3922 1 manual record locking (0 if none)
2653 3923 1 IMPLICIT INPUTS:
2654 3924 1
2655 3925 1 RAB$W_RSZ record size for foreign buffer
2656 3926 1 RAB$L_UBF buffer address for foreign buffer
2657 3927 1
2658 3928 1 IMPLICIT OUTPUTS:
2659 3929 1
2660 3930 1 RAB$B_RAC Record access field
2661 3931 1 RECOUNT Own storage for RECOUNT function.
2662 3932 1 RAB$L_RBF Record pointer in RAB.
2663 3933 1 RAB$W_USZ record size of file buffer
2664 3934 1 RAB$L_UBF address of buffer for file buffer
2665 3935 1
2666 3936 1 ROUTINE VALUE:
2667 3937 1
2668 3938 1 NONE
2669 3939 1
2670 3940 1 SIDE EFFECTS:
2671 3941 1
2672 3942 1 SIGNALS any RMS errors
2673 3943 1 --
2674 3944 1
2675 3945 2 BEGIN
2676 3946 2
2677 3947 2 EXTERNAL REGISTER
2678 3948 2 CCB : REF BLOCK [, BYTE];
2679 3949 2
2680 3950 2 MAP
2681 3951 2 FOREIGN_BUFFER : REF BLOCK [, BYTE];
2682 3952 2
2683 3953 2 LOCAL
2684 3954 2 RMS_STATUS,
2685 3955 2 SAVE_USZ; ! Save the USZ
2686 3956 2
2687 3957 2 !+
2688 3958 2 Save USZ in case it is modified. It is faster to always
2689 3959 2 save and restore it, since that eliminates the test for foreign
2690 3960 2 buffers and single-character mode at the end of this routine.
2691 3961 2 --
2692 3962 2 SAVE_USZ = .CCB [RAB$W_USZ];
2693 3963 2
2694 3964 2 !+
```



```
2695 3965 2  ! Set the record pointer field in the RAB to the user buffer. This is
2696 3966 2  ! done on each element transmission and not just at OPEN because of RMS
2697 3967 2  ! Locate mode.
2698 3968 2  ! Fill the input buffer with Nulls. Basic semantics require null fill if
2699 3969 2  ! the record read is shorter than the buffer.
2700 3970 2  ! Set the record access field in the RAB to sequential. Perform the GET.
2701 3971 2  ! If RMS returns a failure status, signal the error. If the GET is
2702 3972 2  ! successful, then update recount.
2703 3973 2  !-
2704 3974 2
2705 3975 2  IF .FOREIGN_BUFFER NEQ 0
2706 3976 2  THEN
2707 3977 2  BEGIN
2708 3978 2  !+
2709 3979 2  ! There is a foreign buffer. Modify the file buffer to point to the
2710 3980 2  ! buffer associated with the foreign buffer's channel.
2711 3981 2  !-
2712 3982 2  CCB [RAB$L_RBF] = CCB [RAB$L_UBF] = .FOREIGN_BUFFER [RAB$L_UBF];
2713 3983 2  CCB [RAB$W_RSZ] = CCB [RAB$W_USZ] = .FOREIGN_BUFFER [RAB$W_USZ];
2714 3984 2  END
2715 3985 2  ELSE
2716 3986 2  CCB [RAB$L_RBF] = .CCB [RAB$L_UBF];
2717 3987 2
2718 3988 2  CCB [RAB$B_RAC] = RAB$C_RFA;
2719 3989 2
2720 3990 2  !+
2721 3991 2  ! Set bits in RAB ROP without destroying ULK.
2722 3992 2  !-
2723 3993 2
2724 3994 2  CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] OR .LOCK_FLAGS;
2725 3995 2
2726 3996 2  RMS_STATUS = $GET (RAB = .CCB);
2727 3997 2
2728 3998 2  IF .RMS_STATUS EQL RMSS_CONTROLC
2729 3999 2  THEN
2730 4000 2  BASS$SIGNAL_CTRLC ();
2731 4001 2
2732 4002 2  IF NOT .RMS_STATUS
2733 4003 2  THEN
2734 4004 2  BEGIN
2735 4005 2  !+
2736 4006 2  ! We cannot call GET_ERROR because we must restore UBF and USZ.
2737 4007 2  !-
2738 4008 2
2739 4009 2  WHILE (.CCB [RAB$L_STS] EQL RMSS_RSA) DO
2740 4010 2  BEGIN
2741 4011 2  $WAIT (RAB = .CCB);
2742 4012 2  $GET (RAB = .CCB);
2743 4013 2  END;
2744 4014 2
2745 4015 2  END;
2746 4016 2
2747 4017 2  !+
2748 4018 2  ! Turn off bits in the RAB ROP so that subsequent I/O operations can not
2749 4019 2  ! inherit them.
2750 4020 2  !-
2751 4021 2
```



! End of BAS\$\$REC\_GRFA

PC	BB	00000	BAS\$\$REC	GRFA::		
					PUSHR	#*M<R2,R3,R4,R5>
					MOVL	R1, R2
					MOVZWL	32(CCB), SAVE_USZ
					TSTL	FOREIGN_BUFFER
					BEQL	1\$
					MOVL	36(FOREIGN_BUFFER), R1
					MOVL	R1, 36(CCB)
					MOVL	R1, 40(CCB)
					MOVZWL	32(FOREIGN_BUFFER), R0
					MOVW	R0, 32(CCB)
					MOVW	R0, 34(CCB)
					BRB	2\$
					MOVL	36(CCB), 40(CCB)
					MOVB	#2, 30(CCB)
					BISL2	LOCK_FLAGS, 4(CCB)
					PUSHL	CCB
					CALLS	#1, SYSSGET
					MOVL	R0, RMS_STATUS
					CMPL	RMS_STATUS, #67153
					BNEQ	3\$



00000000G	00	00	FB	00049	CALLS	#0, BAS\$\$SIGNAL_CTRL	4000
000182DA	8F	08	AB	D1 00053	BLBS	RMS_STATUS, 5\$	4002
			14	12 0005B	CMPL	8(CCB), #99034	4009
00000000G	00		5B	DD 0005D	BNEQ	5\$	4011
00000000G	00		01	FB 0005F	PUSHL	CCB	4012
			5B	DD 00066	CALLS	#1, SYSS\$WAIT	4009
			01	FB 00068	PUSHL	CCB	4022
			E2	11 0006F	CALLS	#1, SYSS\$GET	4028
04	AB		52	CC 00071	BRB	4\$	4031
22	AB	20	AB	B1 00075	XORL2	LOCK_FLAGS, 4(CCB)	
			1D	1B 0007A	CMPL	32(CCB), 34(CCB)	
			AB	E9 0007C	BLEQU	6\$	
	19	08	AB	3C 00080	BLBC	8(CCB), 6\$	
	51	20	AB	3C 00084	MOVZWL	32(CCB), R1	
	50	22	AB	3C 00088	MOVZWL	34(CCB), R0	
	51		50	C2 0008B	SUBL2	R0, R1	
	50	22	AB	3C 0008F	MOVZWL	34(CCB), R0	
	50	24	AB	C0 00093	ADDL2	36(CCB), R0	
51	00		00	2C 00098	MOVC5	#0, (SP), #0, R1, (R0)	
			60				
	24	AB	9C	AB D0 00099	MOVL	-100(CCB), 36(CCB)	4036
	20	AB	6E	B0 0009E	MOVW	SAVE_USZ, 32(CCB)	4037
00000000	EF	22	AB	3C 000A2	MOVZWL	34(CCB), RECOUNT	4038
	0A	08	AB	E8 000AA	BLBS	8(CCB), 7\$	4044
	7E		01	CE 000AE	MNEGL	#1, -(SP)	
00000000G	00		01	FB 000B1	CALLS	#1, BAS\$\$STOP_IO	
EC	AB	28	AB	D0 000B8	MOVL	40(CCB), -20(CCB)	4050
	5E		04	C0 000BD	ADDL2	#4, SP	4052
			3C	BA 000C0	POPR	#M<R2,R3,R4,R5>	
			05	000C2	RSB		

; Routine Size: 195 bytes, Routine Base: \_BAS\$CODE + 091B

; 2783 4053 1  
; 2784 4054 1

```
: 2786      4055 1 GLOBAL ROUTINE BASS$REC_PSE (      ! PUT (sequential) a record
: 2787      4056 1      COUNT,                      ! No. of bytes to write
: 2788      4057 1      FOREIGN_BUFFER                ! pointer to foreign buffer CB or 0
: 2789      4058 1      ) : JSB_PUT NOVALUE =
: 2790      4059 1
: 2791      4060 1      ++
: 2792      4061 1      FUNCTIONAL DESCRIPTION:
: 2793      4062 1
: 2794      4063 1          Check for "foreign buffers" and point RAB$L_RSZ to foreign USZ if there.
: 2795      4064 1          Write one record. If successful then return; otherwise, signal a fatal
: 2796      4065 1          error.
: 2797      4066 1
: 2798      4067 1      FORMAL PARAMETERS:
: 2799      4068 1
: 2800      4069 1          COUNT.rl.v                No. of bytes to write
: 2801      4070 1          FOREIGN_BUFFER.rl.v        pointer to foreign buffer CB or 0
: 2802      4071 1
: 2803      4072 1      IMPLICIT INPUTS:
: 2804      4073 1
: 2805      4074 1          RAB$W_RSZ                  of foreign buffer
: 2806      4075 1          RAB$L_RBF                  of foreign buffer
: 2807      4076 1          LUB$V_CCO                  Cancel control 0
: 2808      4077 1
: 2809      4078 1      IMPLICIT OUTPUTS:
: 2810      4079 1
: 2811      4080 1          RAB$L_RBF                  for "file" buffer
: 2812      4081 1          RAB$W_RSZ                  length of record to write
: 2813      4082 1          LUB$L_LOG_RECNO             logical record number
: 2814      4083 1          RAB$B_RAC                  record access field
: 2815      4084 1          RAB$V_CCO                  Cancel control 0
: 2816      4085 1
: 2817      4086 1      ROUTINE VALUE:
: 2818      4087 1
: 2819      4088 1          NONE
: 2820      4089 1
: 2821      4090 1      SIDE EFFECTS:
: 2822      4091 1
: 2823      4092 1          SIGNALs any RMS errors
: 2824      4093 1      --
: 2825      4094 1
: 2826      4095 2      BEGIN
: 2827      4096 2
: 2828      4097 2      EXTERNAL REGISTER
: 2829      4098 2          CCB : REF BLOCK [, BYTE];
: 2830      4099 2
: 2831      4100 2      LOCAL
: 2832      4101 2          RMS_STATUS;
: 2833      4102 2
: 2834      4103 2      MAP
: 2835      4104 2          FOREIGN_BUFFER : REF BLOCK [, BYTE];
: 2836      4105 2
: 2837      4106 2      ++
: 2838      4107 2      Copy the current status of the cancel-control-o bit in the LUB
: 2839      4108 2      (possibly set by RCTRL0) into the RAB, and clear it from the
: 2840      4109 2      LUB. The net effect of this is that if the bit is set in the
: 2841      4110 2      LUB, then the CANCTRL0 modifier will be applied to this write
: 2842      4111 2      operation only.
```



```
2843 4112 2 :-
2844 4113 2
2845 4114 2 CCB [RAB$V_CCO] = .CCB [LUB$V_CCO];
2846 4115 2 CCB [LUB$V_CCO] = 0;
2847 4116 2
2848 4117 2
2849 4118 2 +
2850 4119 2 Set the recordsize field in the RAB based on COUNT.
2851 4120 2 Set the record address field in the RAB to the user buffer.
2852 4121 2 Perform the PUT.
2853 4122 2 If RMS returns a failure status, signal the error.
2854 4123 2 -
2855 4124 2 CCB [RAB$W_RSZ] = .COUNT;
2856 4125 2 CCB [RAB$B_RAC] = (IF .CCB [LUB$B_ORGAN] EQL LUB$K_ORG_INDEX THEN RAB$C_KEY ELSE RAB$C_SEQ);
2857 4126 2
2858 4127 2 IF .FOREIGN_BUFFER NEQA 0
2859 4128 2 THEN
2860 4129 2 +
2861 4130 2 There is a foreign buffer. Point RAB$L_UBF to it.
2862 4131 2 -
2863 4132 2 CCB [RAB$L_RBF] = CCB [RAB$L_UBF] = .FOREIGN_BUFFER [RAB$L_UBF]
2864 4133 2 ELSE
2865 4134 2 CCB [RAB$L_RBF] = .CCB [RAB$L_UBF];
2866 4135 2
2867 4136 2 RMS_STATUS = $PUT (RAB = .CCB);
2868 4137 2
2869 4138 2 IF .RMS_STATUS EQL RMSS_CONTROLC
2870 4139 2 THEN
2871 4140 2 BAS$$SIGNAL_CTRLC ();
2872 4141 2
2873 4142 2 IF NOT .RMS_STATUS
2874 4143 2 THEN
2875 4144 2 BEGIN
2876 4145 2 +
2877 4146 2 We cannot call PUT_ERROR because we must restore UBF and USZ.
2878 4147 2 -
2879 4148 2
2880 4149 2 WHILE (.CCB [RAB$L_STS] EQL RMSS_RSA) DO
2881 4150 2 BEGIN
2882 4151 2 $WAIT (RAB = .CCB);
2883 4152 2 $PUT (RAB = .CCB);
2884 4153 2 END;
2885 4154 2
2886 4155 2 END;
2887 4156 2
2888 4157 2 +
2889 4158 2 Restore RAB$L_UBF in case there was a foreign buffer.
2890 4159 2 -
2891 4160 2 CCB [RAB$L_UBF] = .CCB [LUB$A_UBF];
2892 4161 2 +
2893 4162 2 Point LUB$A_RBUF_PTR off to the buffer used by RMS.
2894 4163 2 -
2895 4164 2 CCB [LUB$A_RBUF_ADR] = .CCB [RAB$L_UBF];
2896 4165 2 +
2897 4166 2 Any error remaining (which will be an error other than Record Stream
2898 4167 2 Active, RSA) is fatal.
2899 4168 2 -
```

```
: 2900      4169  2
: 2901      4170  2
: 2902      4171  2
: 2903      4172  2
: 2904      4173  1

IF ( NOT .CCB [RAB$L_STS]) THEN BAS$$STOP_IO (BAS$K_IOERR_REC);

RETURN;
END;

! End of BAS$$REC_PSE
```

07	7E	A0	AB	01	5E	04	C2	00000	BAS\$\$REC_PSE::		
				01		02	EF	00003	SUBL2	#4, SP	: 4055
				07		0E	F0	00009	EXTZV	#2, #1, -96(CCB), -(SP)	: 4114
				AB		04	8A	0000F	INSV	(SP)+, #7, #1, 7(CCB)	
		A0		AB		51	B0	00013	BICB2	#4, -96(CCB)	: 4115
		22		AB		AB	91	00017	MOVW	COUNT, 34(CCB)	: 4124
				03		05	12	0001B	CMPB	-60(CCB), #3	: 4125
						01	D0	0001D	BNEQ	1\$	
				51		02	11	00020	MOVL	#1, R1	
						51	D4	00022	BRB	2\$	
		1E		AB		51	90	00024	CLRL	R1	
						50	D5	00028	MOVB	R1, 30(CCB)	
						0E	13	0002A	TSTL	FOREIGN_BUFFER	: 4127
				50		A0	D0	0002C	BEQL	3\$	
		24		AB		50	D0	00030	MOVL	36(FOREIGN_BUFFER), R0	: 4132
		28		AB		50	D0	00034	MOVL	R0, 36(CCB)	
						05	11	00038	MOVL	R0, 40(CCB)	
		28		AB		AB	D0	0003A	BRB	4\$	
						5B	DD	0003F	MOVL	36(CCB), 40(CCB)	: 4134
				00		01	FB	00041	PUSHL	CCB	: 4136
		00000000G		6E		50	D0	00048	CALLS	#1, SYSSPUT	
		00010651		8F		6E	D1	0004B	MOVL	R0, RMS_STATUS	
						07	12	00052	CMPL	RMS_STATUS, #67153	: 4138
		00000000G		00		00	FB	00054	BNEQ	5\$	
				1E		6E	E8	0005B	CALLS	#0, BAS\$\$SIGNAL_CTRL	: 4140
		000182DA		8F		AB	D1	0005E	BLBS	RMS_STATUS, 7\$	: 4142
						14	12	00066	CMPL	8(CCB), #99034	: 4149
						5B	DD	00068	BNEQ	7\$	
		00000000G		00		01	FB	0006A	PUSHL	CCB	: 4151
						5B	DD	00071	CALLS	#1, SYSSWAIT	
		00000000G		00		01	FB	00073	PUSHL	CCB	: 4152
						E2	11	0007A	CALLS	#1, SYSSPUT	
		24		AB		AB	D0	0007C	BRB	6\$	: 4149
		EC		AB		AB	D0	00081	MOVL	-100(CCB), 36(CCB)	: 4160
				0A		AB	E8	00086	MOVL	36(CCB), -20(CCB)	: 4164
				7E		01	CE	0008A	BLBS	8(CCB), 8\$	: 4170
		00000000G		00		01	FB	0008D	MNEGL	#1, -(SP)	
				5E		04	C0	00094	CALLS	#1, BAS\$\$STOP_IO	
						05	00097	ADDL2	#4, SP		: 4173
								RSB			

; Routine Size: 152 bytes, Routine Base: \_BAS\$CODE + 09DE

; 2905 4174 1



```
2907 4175 1 GLOBAL ROUTINE BAS$$REC_PRE (
2908 4176 1     COUNT,
2909 4177 1     FOREIGN_BUFFER
2910 4178 1 ) : JSB_PUT-NOVALUE =
2911 4179 1
2912 4180 1 ++
2913 4181 1 FUNCTIONAL DESCRIPTION:
2914 4182 1
2915 4183 1     Check for a foreign buffer and point to it if necessary.
2916 4184 1     Write one record. If successful then return; otherwise, signal a fatal
2917 4185 1     error.
2918 4186 1
2919 4187 1 FORMAL PARAMETERS:
2920 4188 1
2921 4189 1     COUNT.rl.v           No. of bytes to write
2922 4190 1     FOREIGN_BUFFER.rl.v  pointer to foreign CB or 0
2923 4191 1
2924 4192 1 IMPLICIT INPUTS:
2925 4193 1
2926 4194 1     RAB$L_UBF             for the foreign buffer (buffer pointer)
2927 4195 1     RAB$W_USZ             for the foreign buffer (buffer size)
2928 4196 1
2929 4197 1 IMPLICIT OUTPUTS:
2930 4198 1
2931 4199 1     RAB$W_RSZ             length of record to write (file buffer)
2932 4200 1     RAB$L_RBF             pointer to file CB
2933 4201 1     LUB$L_LOG_RECNO       logical record number
2934 4202 1     RAB$B_RAC             record access field
2935 4203 1
2936 4204 1 ROUTINE VALUE:
2937 4205 1
2938 4206 1     NONE
2939 4207 1
2940 4208 1 SIDE EFFECTS:
2941 4209 1
2942 4210 1     SIGNALs any RMS errors
2943 4211 1 --
2944 4212 1
2945 4213 2 BEGIN
2946 4214 2
2947 4215 2 EXTERNAL REGISTER
2948 4216 2     CCB : REF BLOCK [, BYTE];
2949 4217 2
2950 4218 2 LOCAL
2951 4219 2     RMS_STATUS;
2952 4220 2
2953 4221 2 MAP
2954 4222 2     FOREIGN_BUFFER : REF BLOCK [, BYTE];
2955 4223 2
2956 4224 2 ++
2957 4225 2     Set the recordsize field in the RAB based on COUNT.
2958 4226 2     Set the record address field in the RAB to the user buffer.
2959 4227 2     Set the record access field in the RAB to relative. Perform the PUT.
2960 4228 2     If RMS returns a failure status, signal the error.
2961 4229 2 --
2962 4230 2
2963 4231 2     CCB [RAB$W_RSZ] = .COUNT;
```

```
2964 4232 2 CCB [RAB$B_RAC] = RAB$C_KEY;
2965 4233 2
2966 4234 2 IF .FOREIGN_BUFFER NEQA 0
2967 4235 2 THEN
2968 4236 2 +
2969 4237 2 | There is a foreign buffer. Point off to the buffer but don't do the
2970 4238 2 | size. A PUT with count would not work right, so the size is passed in.
2971 4239 2 | -
2972 4240 2 CCB [RAB$L_UBF] = CCB [RAB$L_RBF] = .FOREIGN_BUFFER [RAB$L_UBF]
2973 4241 2 ELSE
2974 4242 2 CCB [RAB$L_RBF] = .CCB [RAB$L_UBF];
2975 4243 2
2976 4244 2 RMS_STATUS = $PUT (RAB = .CCB);
2977 4245 2
2978 4246 2 IF .RMS_STATUS EQL RMSS$CONTROL
2979 4247 2 THEN
2980 4248 2 BAS$$SIGNAL_CTRL ();
2981 4249 2
2982 4250 2 IF NOT .RMS_STATUS
2983 4251 2 THEN
2984 4252 2 BEGIN
2985 4253 2 +
2986 4254 2 | We cannot call GET_ERROR because we must restore UBF and USZ.
2987 4255 2 | -
2988 4256 2
2989 4257 2 WHILE (.CCB [RAB$L_STS] EQL RMSS$RSA) DO
2990 4258 2 BEGIN
2991 4259 2 $WAIT (RAB = .CCB);
2992 4260 2 $PUT (RAB = .CCB);
2993 4261 2 END;
2994 4262 2
2995 4263 2 END;
2996 4264 2
2997 4265 2 +
2998 4266 2 | Restore RAB$L_UBF in case there was a foreign buffer.
2999 4267 2 | -
3000 4268 2 CCB [RAB$L_UBF] = .CCB [LUB$A_UBF];
3001 4269 2 +
3002 4270 2 | Point LUB$A_BUF_PTR off to the buffer used by RMS.
3003 4271 2 | -
3004 4272 2 CCB [LUB$A_RBUF_ADR] = .CCB [RAB$L_UBF];
3005 4273 2 +
3006 4274 2 | Any error remaining (which will be an error other than Record Stream
3007 4275 2 | Active, RSA) is fatal.
3008 4276 2 | -
3009 4277 2
3010 4278 2 IF ( NOT .CCB [RAB$L_STS]) THEN BAS$$STOP_IO (BAS$K_IOERR_REC);
3011 4279 2
3012 4280 2 RETURN;
3013 4281 2 END;
! End of BAS$$REC_PRE
```



22	AB		51	B0	00003	MOVW	COUNT, 34(CCB)	4231
1E	AB		01	90	00007	MOVB	#1, 30(CCB)	4232
			50	D5	0000B	TSTL	FOREIGN_BUFFER	4234
	50	24	0E	13	0000D	BEQL	1\$	
28	AB		A0	D0	0000F	MOVL	36(FOREIGN_BUFFER), R0	4240
24	AB		50	D0	00013	MOVL	R0, 40(CCB)	
			50	D0	00017	MOVL	R0, 36(CCB)	
28	AB	24	05	11	0001B	BRB	2\$	
			AB	D0	0001D	MOVL	36(CCB), 40(CCB)	4242
00000000G	00		5B	DD	00022	PUSHL	CCB	4244
	6E		01	FB	00024	CALLS	#1, SYSS\$PUT	
00010651	8F		50	D0	0002B	MOVL	R0, RMS_STATUS	
			6E	D1	0002E	CMPL	RMS_STATUS, #67153	4246
00000000G	00		07	12	00035	BNEQ	3\$	
	1E		00	FB	00037	CALLS	#0, BAS\$\$SIGNAL_CTRL	4248
000182DA	8F	08	6E	E8	0003E	BLBS	RMS_STATUS, 5\$	4250
			AB	D1	00041	CMPL	8(CCB), #99034	4257
			14	12	00049	BNEQ	5\$	
00000000G	00		5B	DD	0004B	PUSHL	CCB	4259
			01	FB	0004D	CALLS	#1, SYSS\$WAIT	
00000000G	00		5B	DD	00054	PUSHL	CCB	4260
			01	FB	00056	CALLS	#1, SYSS\$PUT	
			E2	11	0005D	BRB	4\$	4257
24	AB	9C	AB	D0	0005F	MOVL	-100(CCB), 36(CCB)	4268
EC	AB	24	AB	D0	00064	MOVL	36(CCB), -20(CCB)	4272
	0A	08	AB	E8	00069	BLBS	8(CCB), 6\$	4278
	7E		01	CE	0006D	MNEGL	#1, -(SP)	
00000000G	00		01	FB	00070	CALLS	#1, BAS\$\$STOP_IO	
	5E		04	C0	00077	ADDL2	#4, SP	4281
			05	00	0007A	RSB		

; Routine Size: 123 bytes, Routine Base: \_BAS\$CODE + 0A76

; 3014 4282 1

```

3016 4283 1 GLOBAL ROUTINE BASS$REC_FSE (                ! FIND (sequential) a record
3017 4284 1     LOCK_FLAGS
3018 4285 1 ) : JSB_REC2 NOVALUE =
3019 4286 1
3020 4287 1 !++
3021 4288 1 FUNCTIONAL DESCRIPTION:
3022 4289 1
3023 4290 1     Find next record.  If successful then return; otherwise, signal a fatal
3024 4291 1     error.
3025 4292 1
3026 4293 1 FORMAL PARAMETERS:
3027 4294 1
3028 4295 1     LOCK_FLAGS.rlu.v      bits to set in the RAB ROP to control manual
3029 4296 1                          record locking (0 if none)
3030 4297 1
3031 4298 1 IMPLICIT INPUTS:
3032 4299 1
3033 4300 1     NONE
3034 4301 1
3035 4302 1 IMPLICIT OUTPUTS:
3036 4303 1
3037 4304 1     RAB$B_RAC              record access field
3038 4305 1
3039 4306 1 ROUTINE VALUE:
3040 4307 1
3041 4308 1     NONE
3042 4309 1
3043 4310 1 SIDE EFFECTS:
3044 4311 1
3045 4312 1     Finds next record in file on this logical unit.
3046 4313 1     SIGNALs any RMS errors
3047 4314 1 !--
3048 4315 1
3049 4316 2 BEGIN
3050 4317 2
3051 4318 2 EXTERNAL REGISTER
3052 4319 2     CCB : REF BLOCK [, BYTE];
3053 4320 2
3054 4321 2 LOCAL
3055 4322 2     RMS_STATUS;
3056 4323 2
3057 4324 2 !+
3058 4325 2     Set the record access field in the RAB to sequential.  Perform the FIND.
3059 4326 2     If RMS returns a failure status, signal the error.
3060 4327 2 !-
3061 4328 2
3062 4329 2     CCB [RAB$B_RAC] = RAB$C_SEQ;
3063 4330 2
3064 4331 2 !+
3065 4332 2     Set bits in RAB ROP without clearing ULK.
3066 4333 2 !-
3067 4334 2
3068 4335 2     CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] OR .LOCK_FLAGS;
3069 4336 2
3070 4337 2 !+
3071 4338 2     perform the FIND.
3072 4339 2 !-
```



```
: 3073      4340  2
: 3074      4341  2
: 3075      4342  2
: 3076      4343  2
: 3077      4344  2
: 3078      4345  2
: 3079      4346  2
: 3080      4347  2
: 3081      4348  2
: 3082      4349  2
: 3083      4350  2
: 3084      4351  2
: 3085      4352  2
: 3086      4353  2
: 3087      4354  2
: 3088      4355  2
: 3089      4356  2
: 3090      4357  1

RMS_STATUS = $FIND (RAB = .CCB);

!+
!- Turn off bits so that subsequent operations will not inherit them.
!-

CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] XOR .LOCK_FLAGS;

!+
!- signal if the FIND failed.
!-
IF NOT .RMS_STATUS
THEN
    BAS$$STOP_IO (BAS$K_IOERR_REC);

RETURN;
END;
```

! End of BAS\$\$REC\_FSE

				.EXTRN	SYSS\$FIND	
			52	DD 00000	BAS\$\$REC_FSE::	
					PUSHL	R2
	52		50	D0 00002	MOVL	R0, R2
		1E	AB	94 00005	CLRB	30(CCB)
	04	AB	52	C8 00008	BISL2	LOCK_FLAGS, 4(CCB)
			5B	DD 0000C	PUSHL	CCB
00000000G	00		01	FB 0000E	CALLS	#1, SYSS\$FIND
	04	AB	52	CC 00015	XORL2	LOCK_FLAGS, 4(CCB)
		0A	50	E8 00019	BLBS	RMS_STATUS, 1\$
		7E	01	CE 0001C	MNEGL	#1, -(SP)
00000000G	00		01	FB 0001F	CALLS	#1, BAS\$\$STOP_IO
			04	BA 00026	POPR	#^M<R2>
			05	00028	RSB	

; Routine Size: 41 bytes, Routine Base: \_BAS\$CODE + 0AF1

```
3092 4358 1 GLOBAL ROUTINE BAS$$REC_FRFA (          ! FIND (by RFA) a record
3093 4359 1     LOCK_FLAGS
3094 4360 1 ) : JSB_REC2 NOVALUE =
3095 4361 1
3096 4362 1 ++
3097 4363 1 FUNCTIONAL DESCRIPTION:
3098 4364 1
3099 4365 1     Find record by RFA stored in the RAB.  If successful then return; otherwise, signal a fatal
3100 4366 1     error.
3101 4367 1
3102 4368 1 FORMAL PARAMETERS:
3103 4369 1
3104 4370 1     LOCK_FLAGS.rlu.v          bits to set in the RAB ROP to control manual
3105 4371 1                               record locking (0 if none)
3106 4372 1
3107 4373 1 IMPLICIT INPUTS:
3108 4374 1
3109 4375 1     NONE
3110 4376 1
3111 4377 1 IMPLICIT OUTPUTS:
3112 4378 1
3113 4379 1     RAB$B_RAC                  record access field
3114 4380 1
3115 4381 1 ROUTINE VALUE:
3116 4382 1
3117 4383 1     NONE
3118 4384 1
3119 4385 1 SIDE EFFECTS:
3120 4386 1
3121 4387 1     Finds next record in file on this logical unit.
3122 4388 1     SIGNALs any RMS errors
3123 4389 1 --
3124 4390 1
3125 4391 2 BEGIN
3126 4392 2
3127 4393 2 EXTERNAL REGISTER
3128 4394 2     CCB : REF BLOCK [, BYTE];
3129 4395 2
3130 4396 2 LOCAL
3131 4397 2     RMS_STATUS;
3132 4398 2
3133 4399 2 ++
3134 4400 2     Set the record access field in the RAB to sequential.  Perform the FIND.
3135 4401 2     If RMS returns a failure status, signal the error.
3136 4402 2 --
3137 4403 2
3138 4404 2 CCB [RAB$B_RAC] = RAB$C_RFA;
3139 4405 2
3140 4406 2 ++
3141 4407 2     Set bits in RAB ROP without clearing ULK.
3142 4408 2 --
3143 4409 2
3144 4410 2 CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] OR .LOCK_FLAGS;
3145 4411 2
3146 4412 2 ++
3147 4413 2     perform the FIND.
3148 4414 2 --
```



! End of BAS\$\$REC\_FRFA

: 3167	4433	1
: 3168	4434	1

```

: 3170      4435 1 GLOBAL ROUTINE BAS$$REC_FRE (          ! FIND (relative) a record
: 3171      4436 1     LOCK_FLAGS
: 3172      4437 1 ) : JSB_REC0 NOVALUE =
: 3173      4438 1
: 3174      4439 1 !++
: 3175      4440 1 FUNCTIONAL DESCRIPTION:
: 3176      4441 1
: 3177      4442 1     Find next record.  If successful then return; otherwise, signal a fatal
: 3178      4443 1     error.
: 3179      4444 1
: 3180      4445 1 FORMAL PARAMETERS:
: 3181      4446 1
: 3182      4447 1     LOCK_FLAGS.rlu.v      bits to set in the RAB ROP to control manual
: 3183      4448 1                               record locking (0 if none)
: 3184      4449 1
: 3185      4450 1 IMPLICIT INPUTS:
: 3186      4451 1
: 3187      4452 1     NONE
: 3188      4453 1
: 3189      4454 1 IMPLICIT OUTPUTS:
: 3190      4455 1
: 3191      4456 1     RAB$B_RAC              record access field
: 3192      4457 1
: 3193      4458 1 ROUTINE VALUE:
: 3194      4459 1
: 3195      4460 1     NONE
: 3196      4461 1
: 3197      4462 1 SIDE EFFECTS:
: 3198      4463 1
: 3199      4464 1     SIGNALs any RMS errors
: 3200      4465 1 --
: 3201      4466 1
: 3202      4467 2 BEGIN
: 3203      4468 2
: 3204      4469 2 EXTERNAL REGISTER
: 3205      4470 2     CCB : REF BLOCK [, BYTE];
: 3206      4471 2
: 3207      4472 2 LOCAL
: 3208      4473 2     RMS_STATUS;
: 3209      4474 2
: 3210      4475 2 !+
: 3211      4476 2 ! Set the record access field in the RAB to sequential.  Perform the FIND.
: 3212      4477 2 ! If RMS returns a failure status, signal the error.
: 3213      4478 2 !-
: 3214      4479 2
: 3215      4480 2 CCB [RAB$B_RAC] = RAB$C_KEY;
: 3216      4481 2
: 3217      4482 2 !+
: 3218      4483 2 ! Set bits in the RAB ROP without clearing ULK.
: 3219      4484 2 !-
: 3220      4485 2
: 3221      4486 2 CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] OR .LOCK_FLAGS;
: 3222      4487 2
: 3223      4488 2 !+
: 3224      4489 2 ! perform the FIND.
: 3225      4490 2 !-
: 3226      4491 2

```



```

: 3227      4492  2    RMS_STATUS = $FIND (RAB = .CCB);
: 3228      4493  2
: 3229      4494  2    !+
: 3230      4495  2    ! Turn off bits in the RAB ROP so that subsequent operations do not
: 3231      4496  2    ! inherit them.
: 3232      4497  2    !-
: 3233      4498  2
: 3234      4499  2    CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] XOR .LOCK_FLAGS;
: 3235      4500  2
: 3236      4501  2    !+
: 3237      4502  2    ! Signal if the FIND failed.
: 3238      4503  2    !-
: 3239      4504  2    IF NOT .RMS_STATUS
: 3240      4505  2    THEN
: 3241      4506  2        BAS$$STOP_IO (BAS$K_IOERR_REC);
: 3242      4507  2
: 3243      4508  2    RETURN;
: 3244      4509  1    END;

```

! End of BAS\$\$REC\_FRE

1E	AB	01	90	00000	BAS\$\$REC FRE::		
					MOVB	#1, 30(CCB)	: 4480
04	AB	04	AE	C8 00004	BISL2	LOCK_FLAGS, 4(CCB)	: 4486
			5B	DD 00009	PUSHL	CCB	: 4492
00000000G	00		01	FB 0000B	CALLS	#1, SYS\$FIND	: 4499
04	AB	04	AE	CC 00012	XORL2	LOCK_FLAGS, 4(CCB)	: 4504
	0A		50	E8 00017	BLBS	RMS_STATUS, 1\$	: 4506
	7E		01	CE 0001A	MNEGL	#1, -(SP)	: 4509
00000000G	00		01	FB 0001D	CALLS	#1, BAS\$\$STOP_IO	
			05	00024 1\$:	RSB		

; Routine Size: 37 bytes, Routine Base: \_BAS\$CODE + 0B44

; 3245 4510 1



```
3247 4511 1 GLOBAL ROUTINE BASS$REC_FIN ( ! FIND (indexed) a record
3248 4512 1 KEY_NO, REL_OP, KEY, LOCK_FLAGS) : JSB_REC_IND1 NOVALUE =
3249 4513 1
3250 4514 1 ++
3251 4515 1 FUNCTIONAL DESCRIPTION:
3252 4516 1
3253 4517 1 Find indicated record. If successful then return; otherwise, signal a fatal
3254 4518 1 error.
3255 4519 1
3256 4520 1 FORMAL PARAMETERS:
3257 4521 1
3258 4522 1 KEY_NO.rl.v key of reference
3259 4523 1 REL_OP.rl.v relational operator for key
3260 4524 1 KEY.rt.dx key to search for
3261 4525 1 LOCK_FLAGS.rlu.v bits to set in the RAB ROP to control
3262 4526 1 manual record locking (0 if none)
3263 4527 1
3264 4528 1 IMPLICIT INPUTS:
3265 4529 1
3266 4530 1 NONE
3267 4531 1
3268 4532 1 IMPLICIT OUTPUTS:
3269 4533 1
3270 4534 1 RAB$L_KBF pointer to the desired key value
3271 4535 1 RAB$B_KSZ size of desired key value
3272 4536 1 RAB$M_KGE relational in RAB$L_ROP indicating greater than
3273 4537 1 or equal
3274 4538 1 RAB$M_KGT relational in RAB$L_ROP indicating greater than
3275 4539 1 RAB$B_KRF indicates key of reference
3276 4540 1 RAB$B_RAC record access field
3277 4541 1
3278 4542 1 ROUTINE VALUE:
3279 4543 1
3280 4544 1 NONE
3281 4545 1
3282 4546 1 SIDE EFFECTS:
3283 4547 1
3284 4548 1 See RMS Reference manual for discussion on whether match will be exact,
3285 4549 1 generic, approximate, or generic-approximate.
3286 4550 1 SIGNALS any RMS errors
3287 4551 1 --
3288 4552 1
3289 4553 2 BEGIN
3290 4554 2
3291 4555 2 EXTERNAL REGISTER
3292 4556 2 CCB : REF BLOCK [, BYTE];
3293 4557 2
3294 4558 2 MAP
3295 4559 2 KEY : REF BLOCK [8, BYTE];
3296 4560 2
3297 4561 2 LITERAL
3298 4562 2 K_EQUAL = 0, ! search for key equal
3299 4563 2 K_GREATER_EQUAL = 1, ! search for key GEQ
3300 4564 2 K_GREATER_THAN = 2; ! search for key GTR
3301 4565 2
3302 4566 2 LOCAL
3303 4567 2 RMS_STATUS;
```



```
3304 4568 2
3305 4569 2
3306 4570 2
3307 4571 2
3308 4572 2
3309 4573 2
3310 4574 2
3311 4575 2
3312 4576 2
3313 4577 2
3314 4578 2
3315 4579 2
3316 4580 2
3317 4581 2
3318 4582 2
3319 4583 2
3320 4584 2
3321 4585 2
3322 4586 2
3323 4587 2
3324 4588 2
3325 4589 2
3326 4590 2
3327 4591 2
3328 4592 2
3329 4593 2
3330 4594 2
3331 4595 2
3332 4596 2
3333 4597 2
3334 4598 2
3335 4599 2
3336 4600 2
3337 4601 2
3338 4602 2
3339 4603 2
3340 4604 2
3341 4605 2
3342 4606 2
3343 4607 2
3344 4608 2
3345 4609 2
3346 4610 2
3347 4611 2
3348 4612 2
3349 4613 2
3350 4614 2
3351 4615 2
3352 4616 2
3353 4617 2
3354 4618 2
3355 4619 2
3356 4620 2
3357 4621 2
3358 4622 2
3359 4623 2
3360 4624 2
```

```
!+
! Set the key buffer field, the key size field, the key of reference,
! and the relational bits in the ROP.
! Set the record access field in the RAB to key. Perform the FIND.
! If RMS returns a failure status, signal the error.
!-
```

```
CCB [RAB$B_RAC] = RAB$C_KEY;
CCB [RAB$L_KBF] = .KEY [DSC$A_POINTER];
CCB [RAB$B_KRF] = .KEY_NO;
CCB [RAB$B_KSZ] = (IF .KEY [DSC$B_DTYPE] NEQ DSC$K_DTYPE_P
THEN
    .KEY [DSC$W_LENGTH]
ELSE
    (.KEY [DSC$W_LENGTH]/2 + 1));
```

```
CASE .REL_OP FROM K_EQUAL TO K_GREATER_THAN OF
SET
```

```
    [K_EQUAL] :
        CCB [RAB$V_KGE] = CCB [RAB$V_KGT] = 0;
```

```
    [K_GREATER_EQUAL] :
        BEGIN
            CCB [RAB$V_KGE] = 1;
            CCB [RAB$V_KGT] = 0;
        END;
```

```
    [K_GREATER_THAN] :
        BEGIN
            CCB [RAB$V_KGT] = 1;
            CCB [RAB$V_KGE] = 0;
        END;
```

```
TES;
```

```
!+
! Set bits in the RAB ROP without clearing ULK.
!-
```

```
CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] OR .LOCK_FLAGS;
```

```
!+
! perform the FIND.
!-
```

```
RMS_STATUS = $FIND (RAB = .CCB);
```

```
!+
! Turn off bits in the RAB ROP so that subsequent operations do not
! inherit them.
!-
```

```
CCB [RAB$L_ROP] = .CCB [RAB$L_ROP] XOR .LOCK_FLAGS;
```

```
!+
! Signal if the FIND failed.
```

```
: 3361      4625  2      !-
: 3362      4626  2      IF NOT .RMS_STATUS
: 3363      4627  2      THEN
: 3364      4628  2      BAS$$STOP_10 (BAS$K_IOERR_REC);
: 3365      4629  2
: 3366      4630  2      RETURN;
: 3367      4631  1      END;
```

! End of BAS\$\$REC\_FIN

			52	DD	00000	BAS\$\$REC	FIN::			
							PUSHL	R2		4511
	1E	AB		01	90	00002	MOVB	#1, 30(CCB)		4576
	30	AB	04	A2	D0	00006	MOVL	4(KEY), 48(CCB)		4577
	35	AB		50	90	0000B	MOVB	KEY NO, 53(CCB)		4578
		15	02	A2	91	0000F	CMPB	2(KEY), #21		4579
				05	13	00013	BEQL	1\$		
		52		62	3C	00015	MOVZWL	(KEY), R2		4581
				08	11	00018	BRB	2\$		
		52		62	3C	0001A	MOVZWL	(KEY), R2		4583
		52		02	C6	0001D	DIVL2	#2, R2		
				52	D6	00020	INCL	R2		
	34	AB		52	90	00022	MOVB	R2, 52(CCB)		4579
		52	04	AB	9E	00026	MOVAB	4(CCB), R2		4589
		00		51	CF	0002A	CASEL	REL OP, #0, #2		4585
02							.WORD	4\$-3\$,-		
0018		000D		0006		0002E		5\$-3\$,-		
								6\$-3\$		
	02	A2	40	8F	8A	00034	BICB2	#64, 2(R2)		4589
				10	11	00039	BRB	7\$		
	02	A2		20	88	0003B	BISB2	#32, 2(R2)		4593
	02	A2	40	8F	8A	0003F	BICB2	#64, 2(R2)		4594
				09	11	00044	BRB	8\$		4585
	02	A2	40	8F	88	00046	BISB2	#64, 2(R2)		4599
	02	A2		20	8A	0004B	BICB2	#32, 2(R2)		4600
		62		53	C8	0004F	BISL2	LOCK_FLAGS, (R2)		4608
				5B	DD	00052	PUSHL	CCB		4614
00000000G	00			01	FB	00054	CALLS	#1, SYSS\$FIND		
	62			53	CC	0005B	XORL2	LOCK_FLAGS, (R2)		4621
	0A			50	E8	0005E	BLBS	RMS_STATUS, 9\$		4626
	7E			01	CE	00061	MNEGL	#1, -(SP)		4628
00000000G	00			01	FB	00064	CALLS	#1, BAS\$\$STOP_10		
				04	BA	0006B	POPR	#^M<R2>		4631
				05	00	006D	RSB			

; Routine Size: 110 bytes, Routine Base: \_BAS\$CODE + 0B69

; 3368 4632 1



```

: 3370      4633 1 GLOBAL ROUTINE BAS$$REC_DSE
: 3371      4634 1 : JSB_RECO NOVALUE =
: 3372      4635 1
: 3373      4636 1 !++
: 3374      4637 1 FUNCTIONAL DESCRIPTION:
: 3375      4638 1
: 3376      4639 1 Delete current record. If successful then return; otherwise, signal a fatal
: 3377      4640 1 error.
: 3378      4641 1
: 3379      4642 1 FORMAL PARAMETERS:
: 3380      4643 1
: 3381      4644 1 NONE
: 3382      4645 1
: 3383      4646 1 IMPLICIT INPUTS:
: 3384      4647 1
: 3385      4648 1 NONE
: 3386      4649 1
: 3387      4650 1 IMPLICIT OUTPUTS:
: 3388      4651 1
: 3389      4652 1 RAB$B_RAC record access field
: 3390      4653 1
: 3391      4654 1 ROUTINE VALUE:
: 3392      4655 1
: 3393      4656 1 NONE
: 3394      4657 1
: 3395      4658 1 SIDE EFFECTS:
: 3396      4659 1
: 3397      4660 1 SIGNALS any RMS errors
: 3398      4661 1 --
: 3399      4662 1
: 3400      4663 2 BEGIN
: 3401      4664 2
: 3402      4665 2 EXTERNAL REGISTER
: 3403      4666 2 CCB : REF BLOCK [, BYTE];
: 3404      4667 2
: 3405      4668 2 !+
: 3406      4669 2 Set the record access field in the RAB to sequential. Perform the DELETE.
: 3407      4670 2 If RMS returns a failure status, signal the error.
: 3408      4671 2 !-
: 3409      4672 2
: 3410      4673 2 CCB [RAB$B_RAC] = RAB$C_SEQ;
: 3411      4674 2
: 3412      4675 2 IF NOT $DELETE (RAB = .CCB) THEN BAS$$STOP_IO (BAS$K_IOERR_REC);
: 3413      4676 2
: 3414      4677 2 RETURN;
: 3415      4678 1 END;

```

! End of BAS\$\$REC\_DSE

.EXTRN SYSS\$DELETE

```

1E AB 94 00000 BAS$$REC_DSE::
5B DD 00003 CLR B 30(CCB)
01 FB 00005 PUSH L CCB
50 E8 0000C CALLS #1, SYSS$DELETE
01 CE 0000F BLBS R0, 1$
MNEGL #1, -(SP)
00000000G 00
OA
7E

```

: 4673  
: 4675  
:  
:  
:

BASS\$REC\_PROC  
1-095

G 11  
16-Sep-1984 01:01:12 VAX-11 Bliss-32 V4.0-742  
14-Sep-1984 11:56:35 [BASRTL.SRC]BASRECPRO.B32;1

Page 92  
(34)

00000000G 00

01 FB 00012 CALLS #1, BASS\$STOP\_IO  
05 00019 1\$: RSB

: 4678

; Routine Size: 26 bytes, Routine Base: \_BASS\$CODE + 0BD7

; 3416 4679 1



```
: 3418      4680 1 GLOBAL ROUTINE BAS$$REC_UNL                ! UNLOCK a record
: 3419      4681 1   : JSB_RECO NOVALUE =
: 3420      4682 1
: 3421      4683 1 !++
: 3422      4684 1 FUNCTIONAL DESCRIPTION:
: 3423      4685 1
: 3424      4686 1     Unlock the current record.  If successful or no records locked,
: 3425      4687 1     then return; otherwise, signal a fatal error.
: 3426      4688 1
: 3427      4689 1 FORMAL PARAMETERS:
: 3428      4690 1
: 3429      4691 1     NONE
: 3430      4692 1
: 3431      4693 1 IMPLICIT INPUTS:
: 3432      4694 1
: 3433      4695 1     NONE
: 3434      4696 1
: 3435      4697 1 IMPLICIT OUTPUTS:
: 3436      4698 1
: 3437      4699 1     RAB$B_RAC                record access field
: 3438      4700 1
: 3439      4701 1 ROUTINE VALUE:
: 3440      4702 1
: 3441      4703 1     NONE
: 3442      4704 1
: 3443      4705 1 SIDE EFFECTS:
: 3444      4706 1
: 3445      4707 1     SIGNALs any RMS errors
: 3446      4708 1 --
: 3447      4709 1
: 3448      4710 2 BEGIN
: 3449      4711 2
: 3450      4712 2 EXTERNAL REGISTER
: 3451      4713 2     CCB : REF BLOCK [, BYTE];
: 3452      4714 2
: 3453      4715 2 !+
: 3454      4716 2 ! Set the record access field in the RAB to sequential.  Perform the UNLOCK.
: 3455      4717 2 ! If RMS returns a failure status, signal the error.
: 3456      4718 2 !-
: 3457      4719 2
: 3458      4720 2 CCB [RAB$B_RAC] = RAB$C_SEQ;
: 3459      4721 2
: 3460      4722 2 IF NOT $RELEASE (RAB = .CCB)
: 3461      4723 2 THEN
: 3462      4724 2
: 3463      4725 2     IF .CCB [RAB$L_STS] NEQ RMSS_RNL
: 3464      4726 2     THEN
: 3465      4727 2 !+
: 3466      4728 2 ! An error was returned, check for 'record not locked'.
: 3467      4729 2 !-
: 3468      4730 2     BAS$$STOP_IO (BAS$K_IOERR_REC);
: 3469      4731 2
: 3470      4732 2 RETURN;
: 3471      4733 1 END;                                ! End of BAS$$REC_UNL
```

.EXTRN SYSS\$RELEASE

	1E	AB	94	00000	BAS\$\$REC_UNL::		
		5B	DD	00003	CLRB	30(CCB)	: 4720
00000000G	00	01	FB	00005	PUSHL	CCB	: 4722
		50	E8	0000C	CALLS	#1, SYSS\$RELEASE	:
000181A0	8F	08	AB	D1 0000F	BLBS	R0, 1\$	:
		0A	13	00017	CML	8(CCB), #98720	: 4725
	7E	01	CE	00019	BEQL	1\$	:
00000000G	00	01	FB	0001C	MNEGL	#1, -(SP)	: 4730
		05	00023	1\$:	CALLS	#1, BAS\$\$STOP_IO	:
					RSB		: 4733

; Routine Size: 36 bytes, Routine Base: \_BAS\$CODE + 0BF1

; 3472 4734 1



```
3474 4735 1 GLOBAL ROUTINE BAS$$REC_FEE ! FREE all locked records
3475 4736 1 : JSB_REC0 NOVALUE =
3476 4737 1
3477 4738 1 !++
3478 4739 1 FUNCTIONAL DESCRIPTION:
3479 4740 1
3480 4741 1 Free all locked records. If successful or no records locked,
3481 4742 1 then return; otherwise, signal a fatal error.
3482 4743 1
3483 4744 1 FORMAL PARAMETERS:
3484 4745 1
3485 4746 1 NONE
3486 4747 1
3487 4748 1 IMPLICIT INPUTS:
3488 4749 1
3489 4750 1 NONE
3490 4751 1
3491 4752 1 IMPLICIT OUTPUTS:
3492 4753 1
3493 4754 1 RAB$B_RAC record access field
3494 4755 1
3495 4756 1 ROUTINE VALUE:
3496 4757 1
3497 4758 1 NONE
3498 4759 1
3499 4760 1 SIDE EFFECTS:
3500 4761 1
3501 4762 1 SIGNALs any RMS errors
3502 4763 1 --
3503 4764 1
3504 4765 2 BEGIN
3505 4766 2
3506 4767 2 EXTERNAL REGISTER
3507 4768 2 CCB : REF BLOCK [, BYTE];
3508 4769 2
3509 4770 2 !+
3510 4771 2 Set the record access field in the RAB to sequential. Perform the FREE.
3511 4772 2 If RMS returns a failure status, signal the error.
3512 4773 2 !-
3513 4774 2
3514 4775 2 CCB [RAB$B_RAC] = RAB$C_SEQ;
3515 4776 2
3516 4777 3 IF NOT $FREE (RAB = .CCB)
3517 4778 2 THEN
3518 4779 2
3519 4780 2 IF .CCB [RAB$L_STS] NEQ RMS$_RNL
3520 4781 2 THEN
3521 4782 2 !+
3522 4783 2 An error was returned, check for "record not locked".
3523 4784 2 !-
3524 4785 2 BAS$$STOP_IO (BAS$K_IOERR_REC);
3525 4786 2
3526 4787 2 RETURN;
3527 4788 1 END; ! End of BAS$$REC_FEE
```

BASS\$REC\_PROC  
1-095

K 11  
16-Sep-1984 01:01:12  
14-Sep-1984 11:56:35

VAX-11 Bliss-32 V4.0-742  
[BASRTL.SRC]BASRECPRO.B32;1

Page 96  
(36)

.EXTRN SYSS\$FREE

	1E	AB	94	00000	BASS\$REC FEE::		
		5B	DD	00003	CLRB	30(CCB)	: 4775
00000000G	00	01	FB	00005	PUSHL	CCB	: 4777
		50	E8	0000C	CALLS	#1, SYSS\$FREE	:
000181A0	8F	08	AB	D1 0000F	BLBS	R0, 1\$	:
		0A	13	00017	CMPL	8(CCB), #98720	: 4780
	7E	01	CE	00019	BEQL	1\$	:
00000000G	00	01	FB	0001C	MNEGL	#1, -(SP)	: 4785
		05	00023	1\$:	CALLS	#1, BASS\$STOP_IO	:
					RSB		: 4788

; Routine Size: 36 bytes, Routine Base: \_BASS\$CODE + 0C15

; 3528 4789 1



```

3530 4790 1 GLOBAL ROUTINE BAS$$REC_UPD ( ! UPDATE a record
3531 4791 1 COUNT ! No. of bytes in the record
3532 4792 1 ) : JSB_DO_WRITE NOVALUE =
3533 4793 1
3534 4794 1 !++
3535 4795 1 FUNCTIONAL DESCRIPTION:
3536 4796 1
3537 4797 1 Update current record. If successful then return; otherwise, signal a fatal
3538 4798 1 error.
3539 4799 1
3540 4800 1 FORMAL PARAMETERS:
3541 4801 1
3542 4802 1 COUNT.rl.v No. of bytes in record to update
3543 4803 1
3544 4804 1 IMPLICIT INPUTS:
3545 4805 1
3546 4806 1 NONE
3547 4807 1
3548 4808 1 IMPLICIT OUTPUTS:
3549 4809 1
3550 4810 1 RAB$B_RAC record access field
3551 4811 1 RAB$W_RSZ record size
3552 4812 1
3553 4813 1 ROUTINE VALUE:
3554 4814 1
3555 4815 1 NONE
3556 4816 1
3557 4817 1 SIDE EFFECTS:
3558 4818 1
3559 4819 1 Update current record in file on this logical unit.
3560 4820 1 SIGNALs any RMS errors
3561 4821 1 --
3562 4822 1
3563 4823 2 BEGIN
3564 4824 2
3565 4825 2 EXTERNAL REGISTER
3566 4826 2 CCB : REF BLOCK [, BYTE];
3567 4827 2
3568 4828 2 !+
3569 4829 2 Point RBF to the user buffer.
3570 4830 2 Set the record access field in the RAB to sequential. Perform the UPDATE.
3571 4831 2 If RMS returns a failure status, signal the error.
3572 4832 2 --
3573 4833 2 CCB [RAB$L_RBF] = .CCB [RAB$L_UBF];
3574 4834 2 CCB [RAB$W_RSZ] = .COUNT;
3575 4835 2 CCB [RAB$B_RAC] = RAB$C_SEQ;
3576 4836 2
3577 4837 2 IF NOT $UPDATE (RAB = .CCB) THEN BAS$$STOP_IO (BAS$K_ICERR_REC);
3578 4838 2
3579 4839 2 !+
3580 4840 2 Point LUB$A_RBUF_ADR to the buffer used by RMS for MOVE.
3581 4841 2 --
3582 4842 2 CCB [LUB$A_RBUF_ADR] = .CCB [RAB$L_UBF];
3583 4843 2 RETURN;
3584 4844 1 END; ! End of BAS$$REC_UPD

```

				.EXTRN SYSSUPDATE		
28	AB	24	AB	D0 00000	BAS\$\$REC UPD::	
					MOVL	36(CCB), 40(CCB)
22	AB		50	B0 00005	MOVW	COUNT, 34(CCB)
		1E	AB	94 00009	CLRB	30(CCB)
			5B	DD 0000C	PUSHL	CCB
00000000G	00		01	FB 0000E	CALLS	#1, SYSSUPDATE
	0A		50	E8 00015	BLBS	R0, 1\$
	7E		01	CE 00018	MNEGL	#1, -(SP)
00000000G	00		01	FB 0001B	CALLS	#1, BAS\$\$STOP IO
EC	AB	24	AB	D0 00022 1\$:	MOVL	36(CCB), -20(CCB)
			05	00027	RSB	

; Routine Size: 40 bytes, Routine Base: \_BAS\$CODE + 0C39

; 3585 4845 1



! RESTORE (sequential) to beginning of file

```
3587 4846 1 GLOBAL ROUTINE BAS$$REC_RSE
3588 4847 1 : JSB_RECO NOVALUE =
3589 4848 1
3590 4849 1 ++
3591 4850 1 FUNCTIONAL DESCRIPTION:
3592 4851 1
3593 4852 1     Rewind the file.  If successful then return; otherwise, signal a fatal
3594 4853 1     error.
3595 4854 1
3596 4855 1 FORMAL PARAMETERS:
3597 4856 1
3598 4857 1     NONE
3599 4858 1
3600 4859 1 IMPLICIT INPUTS:
3601 4860 1
3602 4861 1     NONE
3603 4862 1
3604 4863 1 IMPLICIT OUTPUTS:
3605 4864 1
3606 4865 1     RAB$B_RAC                record access field
3607 4866 1
3608 4867 1 ROUTINE VALUE:
3609 4868 1
3610 4869 1     NONE
3611 4870 1
3612 4871 1 SIDE EFFECTS:
3613 4872 1
3614 4873 1     SIGNALs any RMS errors
3615 4874 1 --
3616 4875 1
3617 4876 2 BEGIN
3618 4877 2
3619 4878 2 EXTERNAL REGISTER
3620 4879 2     CCB : REF BLOCK [, BYTE];
3621 4880 2
3622 4881 2 ++
3623 4882 2     Set the record access field in the RAB to sequential.  Perform the REWIND.
3624 4883 2     If RMS returns a failure status, signal the error.
3625 4884 2 --
3626 4885 2
3627 4886 2 CCB [RAB$B_RAC] = RAB$C_SEQ;
3628 4887 2
3629 4888 2 IF NOT $REWIND (RAB = .CCB) THEN BAS$$STOP_IO (BAS$K_IOERR_REC);
3630 4889 2
3631 4890 2 RETURN;
3632 4891 1 END;
                                     ! End of BAS$$REC_RSE
```

.EXTRN SYSSREWIND

```
1E AB 94 0000 BAS$$REC_RSE::
                                CLR B 30(CCB)
                                PUSHL CCB
                                CALLS #1, SYSSREWIND
                                BLBS R0, 1$
                                MNEGL #1, -(SP)
```

00000000G	00	5B DD 00003
	0A	01 FB 00005
	7E	50 E8 0000C
		01 CE 0000F

: 4886  
: 4888  
:  
:

BAS\$\$REC\_PROC  
1-095

B 12  
16-Sep-1984 01:01:12  
14-Sep-1984 11:56:35

VAX-11 Bliss-32 V4.0-742  
[BASRTL.SRC]BASRECPRO.B32;1

Page 100  
(38)

00000000G 00

01 FB 00012  
05 00019 1\$:

CALLS #1, BAS\$\$STOP\_IO  
RSB

: 4891

; Routine Size: 26 bytes, Routine Base: \_BAS\$CODE + 0C61

; 3633 4892 1



```
3635 4893 1 GLOBAL ROUTINE BAS$$REC_RIN (                ! RESTORE (indexed) to beginning of file
3636 4894 1     KEY_NO) : JSB_REC_IND NOVALUE =
3637 4895 1
3638 4896 1 ++
3639 4897 1 FUNCTIONAL DESCRIPTION:
3640 4898 1
3641 4899 1     Rewind the file.  If successful then return; otherwise, signal a fatal
3642 4900 1     error.
3643 4901 1
3644 4902 1 FORMAL PARAMETERS:
3645 4903 1
3646 4904 1     KEY_NO.rl.v                                key of reference
3647 4905 1
3648 4906 1 IMPLICIT INPUTS:
3649 4907 1
3650 4908 1     NONE
3651 4909 1
3652 4910 1 IMPLICIT OUTPUTS:
3653 4911 1
3654 4912 1     RAB$B_KRF                                key of reference
3655 4913 1     RAB$B_RAC                                record access field
3656 4914 1
3657 4915 1 ROUTINE VALUE:
3658 4916 1
3659 4917 1     NONE
3660 4918 1
3661 4919 1 SIDE EFFECTS:
3662 4920 1
3663 4921 1     SIGNALs any RMS errors
3664 4922 1 --
3665 4923 1
3666 4924 2 BEGIN
3667 4925 2
3668 4926 2 EXTERNAL REGISTER
3669 4927 2     CCB : REF BLOCK [, BYTE];
3670 4928 2
3671 4929 2 ++
3672 4930 2     Set the key of reference .
3673 4931 2     Set the record access field in the RAB to key.  Perform the REWIND.
3674 4932 2     If RMS returns a failure status, signal the error.
3675 4933 2     --
3676 4934 2
3677 4935 2     CCB [RAB$B_KRF] = .KEY_NO;
3678 4936 2     CCB [RAB$B_RAC] = RAB$C_KEY;
3679 4937 2
3680 4938 2 IF NOT $REWIND (RAB = .CCB) THEN BAS$$STOP_IO (BAS$K_IOERR_REC);
3681 4939 2
3682 4940 2 RETURN;
3683 4941 1 END;                                ! End of BAS$$REC_RIN
```

```
35 AB          50 90 00000 BAS$$REC_RIN::
1E AB          01 90 00004      MOVB KEY_NO, 53(CCB)
                                MOVB #1, -30(CCB)
```

```
: 4935
: 4936
```

Page 102  
(39)[illegible]

: 3684 4942 1



```

: 3686      4943 1 GLOBAL ROUTINE BAS$$REC_SSE      ! SCRATCH (sequential) a record
: 3687      4944 1   : JSB_RECO NOVALUE =
: 3688      4945 1
: 3689      4946 1 ++
: 3690      4947 1 FUNCTIONAL DESCRIPTION:
: 3691      4948 1
: 3692      4949 1     Truncate this file.  If successful then return; otherwise, signal a fatal
: 3693      4950 1     error.
: 3694      4951 1
: 3695      4952 1 FORMAL PARAMETERS:
: 3696      4953 1
: 3697      4954 1     NONE
: 3698      4955 1
: 3699      4956 1 IMPLICIT INPUTS:
: 3700      4957 1
: 3701      4958 1     NONE
: 3702      4959 1
: 3703      4960 1 IMPLICIT OUTPUTS:
: 3704      4961 1
: 3705      4962 1     RAB$B_RAC      record access field
: 3706      4963 1
: 3707      4964 1 ROUTINE VALUE:
: 3708      4965 1
: 3709      4966 1     NONE
: 3710      4967 1
: 3711      4968 1 SIDE EFFECTS:
: 3712      4969 1
: 3713      4970 1     SIGNALs any RMS errors
: 3714      4971 1 --
: 3715      4972 1
: 3716      4973 2 BEGIN
: 3717      4974 2
: 3718      4975 2 EXTERNAL REGISTER
: 3719      4976 2     CCB : REF BLOCK [, BYTE];
: 3720      4977 2
: 3721      4978 2 ++
: 3722      4979 2     Set the record access field in the RAB to sequential.  Perform the TRUNCATE.
: 3723      4980 2     If RMS returns a failure status, signal the error.
: 3724      4981 2 --
: 3725      4982 2
: 3726      4983 2 CCB [RAB$B_RAC] = RAB$C_SEQ;
: 3727      4984 2
: 3728      4985 2 IF NOT $TRUNCATE (RAB = .CCB) THEN BAS$$STOP_IO (BAS$K_IOERR_REC);
: 3729      4986 2
: 3730      4987 2 RETURN;
: 3731      4988 1 END;      ! End of BAS$$REC_SSE
```

.EXTRN SYS\$TRUNCATE

```

1E AB 94 00000 BAS$$REC_SSE::
      5B DD 00003 CLR B 30(CCB)
      01 FB 00005 PUSHL CCB
      50 E8 0000C CALLS #1, SYS$TRUNCATE
      01 CE 0000F BLBS R0, 1$
      MNEGL #1, -(SP)
```

```

00000000G 00
          0A
          7E
```

```

: 4983
: 4985
:
:
```

BASS\$REC\_PROC  
1-095

F 12  
16-Sep-1984 01:01:12  
14-Sep-1984 11:56:35

VAX-11 Bliss-32 V4.0-742  
[BASRTL.SRC]BASRECPRO.B32;1

Page 104  
(40)

00000000G 00

01 FB 00012  
05 00019 1\$:

CALLS #1, BASS\$STOP\_IO  
RSB

: 4988

; Routine Size: 26 bytes, Routine Base: \_BASS\$CODE + 0C9A

; 3732 4989 1



```

: 3734 4990 1 ROUTINE PUT_ERROR (
: 3735 4991 1     SIGNAL_OR_STOP
: 3736 4992 1     ) : CALL_CCB NOVALUE =
: 3737 4993 1
: 3738 4994 1 !++
: 3739 4995 1 ! FUNCTIONAL DESCRIPTION:
: 3740 4996 1
: 3741 4997 1     Here on $PUT errors, check for Record stream active error (RMS$_RSA)
: 3742 4998 1     If this error, WAIT until not active and try $PUT again.
: 3743 4999 1     This recovers from AST I/O which can occur out of the middle
: 3744 5000 1     of synchronous I/O at non-AST level.
: 3745 5001 1
: 3746 5002 1 ! CALLING SEQUENCE:
: 3747 5003 1
: 3748 5004 1     PUT_ERROR (signal_or_stop)
: 3749 5005 1
: 3750 5006 1 ! FORMAL PARAMETERS:
: 3751 5007 1
: 3752 5008 1     SIGNAL_OR_STOP.rl.v           whether to signal or stop
: 3753 5009 1
: 3754 5010 1 ! IMPLICIT INPUTS:
: 3755 5011 1
: 3756 5012 1     CCB                       Adr. of current LUB/ISB/RAB
: 3757 5013 1
: 3758 5014 1 ! IMPLICIT OUTPUTS:
: 3759 5015 1
: 3760 5016 1     LUB$V_OUTBUF_DR           Cleared to indicate clean buffer
: 3761 5017 1
: 3762 5018 1 ! ROUTINE VALUE:
: 3763 5019 1
: 3764 5020 1     NONE
: 3765 5021 1
: 3766 5022 1 ! SIDE EFFECTS:
: 3767 5023 1
: 3768 5024 1     $WAITs and then tries $PUT again, until success or any error
: 3769 5025 1     except record stream active.
: 3770 5026 1 --
: 3771 5027 1
: 3772 5028 2 BEGIN
: 3773 5029 2
: 3774 5030 2 EXTERNAL REGISTER
: 3775 5031 2     CCB : REF BLOCK [, BYTE];
: 3776 5032 2
: 3777 5033 2 WHILE .CCB [RAB$L_STS] EQL RMS$_RSA DO
: 3778 5034 3     BEGIN
: 3779 5035 3         $WAIT (RAB = .CCB);
: 3780 5036 4         $PUT (RAB = .CCB)
: 3781 5037 2     END;
: 3782 5038 2
: 3783 5039 2 IF NOT .CCB [RAB$L_STS]
: 3784 5040 2 THEN
: 3785 5041 2     BEGIN
: 3786 5042 3 !+
: 3787 5043 3 ! Clear the buffer dirty bit so if there is anything there BAS$CLOSE won't
: 3788 5044 3 ! get confused, and try to do another PUT.
: 3789 5045 3 --
: 3790 5046 3     CCB [LUB$V_OUTBUF_DR] = 0;

```

```
: 3791      5047      3
: 3792      5048
: 3793      5049
: 3794      5050      IF .SIGNAL_OR_STOP EQL K_SIGNAL
: 3795      5051      THEN
: 3796      5052      BAS$$SIGNAL_IO (BAS$K_IOERR_REC)
: 3797      5053      ELSE
: 3798      5054      BAS$$STOP_IO (BAS$K_IOERR_REC);
: 3799      5055      END;
: 3800      5056      RETURN;
                        ! End of PUT_ERROR
                        END;
```

```
0000 00000 PUT_ERROR:
000182DA 8F      08 AB D1 00002 1$: .WORD      Save nothing      : 4990
      14 12 0000A      CMPL      8(CCB), #99034      : 5033
      5B DD 0000C      BNEQ      2$
00000000G 00      01 FB 0000E      PUSHL     CCB      : 5035
      5B DD 00015      CALLS     #1, SYSSWAIT
00000000G 00      01 FB 00017      PUSHL     CCB      : 5036
      E2 11 0001E      CALLS     #1, SYSSPUT
      1F      08 AB E8 00020 2$: BRB      1$
      FE AB      08 08 8A 00024      BLBS     8(CCB), 4$      : 5039
      01      04 AC D1 00028      BICB2     #8, -2(CCB)      : 5046
      0B 12 0002C      CMPL     SIGNAL_OR_STOP, #1      : 5048
      01 CE 0002E      BNEQ     3$
00000000G 00      01 FB 00031      MNEGL     #1, -(SP)      : 5050
      04 00038      CALLS     #1, BAS$$SIGNAL_IO
      7E      01 CE 00039 3$: RET
00000000G 00      01 FB 0003C      MNEGL     #1, -(SP)      : 5052
      04 00043 4$: CALLS     #1, BAS$$STOP_IO
      RET      : 5056
```

; Routine Size: 68 bytes, Routine Base: \_BAS\$CODE + 0CB4



```
3802 5057 1 ROUTINE GET_ERROR (
3803 5058 1     SIGNAL_OR_STOP
3804 5059 1     ) : CALL_CCB NOVALUE =
3805 5060 1
3806 5061 1 ++
3807 5062 1 FUNCTIONAL DESCRIPTION:
3808 5063 1
3809 5064 1     Here on $GET errors, check for Record stream active error (RMS$_RSA)
3810 5065 1     If this error, WAIT until not active and try $GET again.
3811 5066 1     This recovers from AST I/O which can occur out of the middle
3812 5067 1     of synchronous I/O at non-AST level.
3813 5068 1
3814 5069 1 CALLING SEQUENCE:
3815 5070 1
3816 5071 1     JSB GET_ERROR ( )
3817 5072 1
3818 5073 1 FORMAL PARAMETERS:
3819 5074 1
3820 5075 1     NONE
3821 5076 1
3822 5077 1 IMPLICIT INPUTS:
3823 5078 1
3824 5079 1     CCB                      Adr. of current LUB/ISB/RAB
3825 5080 1
3826 5081 1 IMPLICIT OUTPUTS:
3827 5082 1
3828 5083 1 ROUTINE VALUE:
3829 5084 1
3830 5085 1     NONE
3831 5086 1
3832 5087 1 SIDE EFFECTS:
3833 5088 1
3834 5089 1     If this is an INPUT LINE and a ^Z error, then just return and it will
3835 5090 1     be handled above.
3836 5091 1     $WAITS and then tries $GET again, until success or any error
3837 5092 1     except record stream active.
3838 5093 1 --
3839 5094 1
3840 5095 2 BEGIN
3841 5096 2
3842 5097 2 EXTERNAL REGISTER
3843 5098 2     CCB : REF BLOCK [, BYTE];
3844 5099 2
3845 5100 2 +
3846 5101 2 Set the prompt buffer length to zero so that error followed by RESUME will not
3847 5102 2 keep concatenating the prompt buffer.
3848 5103 2 -
3849 5104 2     CCB [RAB$_PSZ] = 0;
3850 5105 2 +
3851 5106 2 Is this INPUT LINE and only a ^Z in the record?
3852 5107 2 -
3853 5108 2
3854 5109 2 IF .CCB [ISB$_STIM TYPE] EQL ISB$_ST_TY_INL AND .CCB [RAB$_RSZ] EQLU 1 AND .(.CCB [RAB$_RBF])<0, 8>
3855 5110 2     EQLU BAS$_CONTROL_2
3856 5111 2 THEN
3857 5112 2     RETURN;
3858 5113 2
```

```

: 3859      5114 2  WHILE .CCB [RAB$L_STS] EQL RMSS_RSA DO
: 3860      5115 3  BEGIN
: 3861      5116 3  $WAIT (RAB = .CCB);
: 3862      5117 4  $GET (RAB = .CCB)
: 3863      5118 2  END;
: 3864      5119 2
: 3865      5120 2  IF NOT .CCB [RAB$L_STS]
: 3866      5121 2  THEN
: 3867      5122 2
: 3868      5123 2
: 3869      5124 2  !+ Check the input parameter to see if we should signal or stop.
: 3870      5125 2  !-
: 3871      5126 2
: 3872      5127 2  IF .SIGNAL_OR_STOP EQL K_SIGNAL
: 3873      5128 2  THEN
: 3874      5129 2  BAS$$SIGNAL_IO (BAS$K_IOERR_REC)
: 3875      5130 2  ELSE
: 3876      5131 2  BAS$$STOP_IO (BAS$K_IOERR_REC);
: 3877      5132 2
: 3878      5133 2  RETURN;
: 3879      5134 1  END;

```

! End of GET\_ERROR

0000 00000 GET_ERROR:						
		34	AB 94 00002	.WORD	Save nothing	: 5057
			CB 91 00005	CLRB	52(CCB)	: 5104
	20	FF71	OC 12 0000A	CMPB	-143(CCB), #32	: 5109
	01	22	AB B1 0000C	BNEQ	1\$	
			06 12 00010	CMPW	34(CCB), #1	
	1A	28	BB 91 00012	BNEQ	1\$	: 5110
			3D 13 00016	CMPB	240(CCB), #26	
000182DA	8F	08	AB D1 00018 1\$:	BEQL	4\$	: 5114
			14 12 00020	CMPB	8(CCB), #99034	
			5B DD 00022	BNEQ	2\$	: 5116
00000000G	00		01 FB 00024	PUSHL	CCB	: 5117
			5B DD 0002B	CALLS	#1, SYSSWAIT	
00000000G	00		01 FB 0002D	PUSHL	CCB	
			E2 11 00034	CALLS	#1, SYSSGET	
	1B	08	AB E8 00036 2\$:	BRB	1\$	: 5120
	01	04	AC D1 0003A	BLBS	8(CCB), 4\$	: 5127
			0B 12 0003E	CMPB	SIGNAL_OR_STOP, #1	
	7E		01 CE 00040	BNEQ	3\$	: 5129
00000000G	00		01 FB 00043	MNEGL	#1, -(SP)	
			04 0004A	CALLS	#1, BAS\$\$SIGNAL_IO	
	7E		01 CE 0004B 3\$:	RET		: 5131
00000000G	00		01 FB 0004E	MNEGL	#1, -(SP)	
			04 00055 4\$:	CALLS	#1, BAS\$\$STOP_IO	: 5134
				RET		

; Routine Size: 86 bytes, Routine Base: \_BAS\$CODE + 0CF8

```

: 3880      5135 1 END
: 3881      5136 1

```



BAS\$\$REC\_PROC  
1-095

K 12  
16-Sep-1984 01:01:12  
14-Sep-1984 11:56:35

VAX-11 Bliss-32 V4.0-742  
[BASRTL.SRC]BASRECPRO.B32;1

Page 109  
(42)

; 3882            5137 0 ELUDOM

BAS\$\$REC\_WF9==      BAS\$\$REC\_WSL9  
BAS\$\$REC\_WF1==      BAS\$\$REC\_WSL1  
BAS\$\$REC\_WF0==      BAS\$\$REC\_WSL0

PSECT SUMMARY

Name	Bytes	Attributes
:_BAS\$DATA	6	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, PIC, ALIGN(2)
:_BAS\$CODE	3406	NOVEC, NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC, ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
:_\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	44	0	581	00:01.2

COMMAND QUALIFIERS

; BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS\$:BASRECPRO/OBJ=OBJ\$:BASRECPRO MSRC\$:BASRECPRO/UPDATE=(ENH\$:BASRECPRO  
; )

; Size:            3397 code + 15 data bytes  
; Run Time:        01:16.8  
; Elapsed Time:    02:50.9  
; Lines/CPU Min:   4014  
; Lexemes/CPU-Min: 26026  
; Memory Used:    249 pages  
; Compilation Complete



0030 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

